EMNLP-CoNLL 2007

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning

June 28–30, 2007
Prague, Czech Republic

The Association for Computational Linguistics

Production and Manufacturing by
Omnipress
2600 Anderson Street
Madison, WI 53704
USA

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

Preface

Welcome to EMNLP-CoNLL 2007, an unprecedented joint meeting of the Conference on Empirical Methods in Natural Language Processing (EMNLP) and the Conference on Computational Natural Language Learning (CoNLL).

The conference is a joint effort of SIGDAT and SIGNLL, the ACL Special Interest Groups that usually organize the annual EMNLP and CoNLL conferences, respectively.

Our field is growing rapidly. This year, EMNLP-CoNLL considered a remarkable 398 submissions,[1] accepting 109 of them (for an acceptance rate of 27%). It is startling to realize that even the ACL conferences were not this large until two years ago.

Only 66 of the accepted papers were scheduled for presentation as talks, and 43 more as posters. We took pains to ensure that the poster sessions would be leisurely and interactive.

In addition, two sessions of the conference and 22 specially designated short papers in this volume are devoted to the CoNLL Shared Task competition, an annual tradition. The 2007 competition concerns dependency parsing, with both a multilingual track and a domain adaptation track.

Several innovations this year have received positive feedback and are worth mentioning:

- To encourage thorough citation of related work, a paper's References section was *not* counted against the 8-page limit for submitted papers or the 9-page limit for camera-ready papers.

  (Note that authors were allowed an extra page in the camera-ready version to help them effectively address reviewers' comments, following an innovation at EMNLP 2006.)

- The review form was redesigned (starting from the fine review form of EACL 2006) to provide clearer and more consistent guidance to reviewers, area chairs, and authors. Authors were directed to consult the review form, which was posted at the conference website, while preparing their submissions and when interpreting their review scores.

- Some of our submissions (fewer than 1/3) appeared to be revisions of rejected ACL 2007 submissions. Where possible, we tried to conserve valuable information and effort from the ACL 2007 reviewing process by re-assigning one, though only one, of the ACL reviewers to such a paper.

  Such re-reviewers were instructed to give the new, revised submission the fresh reading that it deserved, but they were also encouraged to bring up points that still applied from any of the ACL 2007 reviews or discussion.

[1] Of the original 419 submissions, 17 were withdrawn (usually upon acceptance elsewhere), and 4 more were rejected without review (for violating the conference's standards on length, anonymization, or plagiarism).

- By accepting many posters and presenting them all *simultaneously*, we hoped to accommodate a large audience without overcrowding at each poster.

  The large number of posters in turn required a long period for poster viewing. With a total of 5 hours spanning two receptions, a conferencegoer can engage with nearly half of the posters for 15 minutes of personalized discussion each. This makes the posters roughly as visible as the talks, which are split into parallel sessions.

- In addition to the Best Paper Award (see Session 1), we are considering organizing—if logistically feasible—an "Audience Choice" award for the most worthwhile *presentation* at the conference. Such a prize would reward authors who not only produced outstanding research but also communicated it clearly and enjoyably at the conference meeting.

It is my privilege to thank the many individuals—most of them listed on the following pages—whose generous efforts have made this conference possible. Foremost are the 16 dedicated area chairs and 370 reviewers, who worked together hard and thoughtfully to select this excellent program and provide valuable feedback to the authors. Also as part of the technical program, Joakim Nivre chaired the organization of the CoNLL Shared Task and the resulting short papers; Taku Kudo ably identified ACL 2007 resubmissions (see above); and Hal Daumé III kindly chaired the best paper award committee. Eric Ringger put a great deal of effort into producing this fine proceedings volume, with support from Su Jian, the ACL publications chair. Jan Hajic coordinated the many local arrangements, along with Priscilla Rasmussen, Anna Kotesovcova, Jiri Mirovsky, Pavel Stranak, Zdenek Zabokrtsky, and no doubt others; we are very grateful to them for making everything run smoothly in Prague. Antal van den Bosch, Dan Jurafsky, Eric Gaussier, and Ken Church provided much valuable advice over the past months based on their experience. Finally, let us not forget the hundreds of authors who actually produced the excellent research in this volume, and the invited speakers who graciously traveled a long way to enlighten us.

Enjoy the conference!

Jason Eisner
EMNLP-CoNLL Chair
May 2007

Organizers

Program Chair:

Jason Eisner, Johns Hopkins University

Area Chairs:

David Chiang, USC Information Sciences Institute
Alexander Clark, Royal Holloway University of London
Michael Collins, Massachusetts Institute of Technology
Hal Daumé III, University of Utah
Sanjeev Khudanpur, Johns Hopkins University
Katrin Kirchhoff, University of Washington
Dekang Lin, Google Inc.
Manabu Okumura, Tokyo Institute of Technology
Anoop Sarkar, Simon Fraser University
Suzanne Stevenson, University of Toronto
Tomek Strzalkowski, SUNY Albany
Carlo Strapparava, Istituto Trentino di Cultura
Hans Uszkoreit, Saarland University DFKI
Menno van Zaanen, Macquarie University
Marilyn Walker, University of Sheffield
Janyce Wiebe, University of Pittsburgh

CoNLL Shared Task Organizers:

Joakim Nivre, Växjö University and Uppsala University (chair)
Johan Hall, Växjö University
Sandra Kübler, Indiana University
Ryan McDonald, Google Inc.
Jens Nilsson, Växjö University
Sebastian Riedel, University of Edinburgh
Deniz Yuret, Koç University

Local Arrangements Chair:

Jan Hajic, Charles University

Publications Chair:

Eric Ringger, Brigham Young University

Reviewers:

John Aberdeen, Pieter Adriaans, Eugene Agichtein, Eneko Agirre, Gregory Aist, Enrique Alfonseca, Rie Ando, Necip Fazil Ayan, Leif Azzopardi;

Srinivas Bangalore, Marco Baroni, Regina Barzilay, Roberto Basili, Sugato Basu, Ron Bekkerman, Anja Belz, Sabine Bergler, Shane Bergsma, Dan Bikel, Misha Bilenko, Dave Blei, John Blitzer, Rens Bod, Bernd Bohnet, Johan Bos, Thorsten Brants, Eric Breck, Chris Brew, David Brooks, Charles Brown, Wray Buntine, John Burger;

Janet Cahn, Giuseppe Carenini, Xavier Carreras Pérez, Neus Català, Damir Ćavar, Joyce Chai, Yee Seng Chan, Jason Chang, Ciprian Chelba, Hsin-Hsi Chen, Stanley Chen, Colin Cherry, Yejin Choi, Jennifer Chu-Carroll, Grace Chung, Ken Church, Massimiliano Ciaramita, Alexander Clark, Stephen Clark, John Coleman, Paul Cook, Christophe Costa Florêncio, Mathias Creutz, Dan Cristea, András Csomai, Elsa Cubel, Silviu Cucerzan, Aron Culotta, James Curran;

Walter Daelemans, Robert Dale, R. I. Damper, Hal Daumé III, Eric Villemonte de la Clergerie, Maarten de Rijke, Christy Doran, Mark Dras, Amit Dubey, Kevin Duh;

Phil Edmonds, Noémie Elhadad, T. Mark Ellison, Ahmad Emami, Katrin Erk, David Evans, Rémi Eyraud;

Afsaneh Fazly, Marcello Federico, Karim Filali, Jenny Finkel, Kate Forbes-Riley, Eric Fosler-Lussier, George Foster, Mary Ellen Foster, Alexander Fraser, Dayne Freitag, Atsushi Fujii, Sean Fulop, Pascale Fung;

Tamas Gaal, Evgeniy Gabrilovich, Michel Galley, Michael Gamon, Claire Gardent, Jeroen Geertzen, Dale Gerdemann, Ulrich Germann, Daniel Gildea, Roxana Girju, John Goldsmith, Jade Goldstein, Sharon Goldwater, Cyril Goutte, Mark Greenwood, Gregory Grefenstette, Tom Griffiths, Iryna Gurevych;

Kadri Hacioglu, Aria Haghighi, Udo Hahn, Jan Hajic, Dilek Hakkani-Tür, Keith Hall, Susan Haller, Hilda Hardy, Mary Harper, Mary Hearne, Marti Hearst, Peter Heeman, James Henderson, John Henderson, Mark Hepple, Ryuichiro Higashinaka, Tsutomu Hirao, Graeme Hirst, Julia Hockenmaier, Chu-Ren Huang, Liang Huang, Annette Hulth, Rebecca Hwa;

Diana Inkpen, Kentaro Inui, Abe Ittycheriah;

Martin Jansche, Valentin Jijkoun, Mark Johnson, Kristiina Jokinen;

Min-Yen Kan, Hiroshi Kanayama, Damianos Karakos, Lauri Karttunen, Rohit Kate, Frank Keller, André Kempe, Adam Kilgarriff, Soo-Min Kim, George Kiraz, Katrin Kirchhoff, Chunyu Kit, Kevin Knight, Alistair Knott, Philipp Koehn, Rob Koeling, Alexander Koller, Grzegorz Kondrak, Stasinos Konstantopoulos, Terry Koo, Moshe Koppel, Anna Korhonen, András Kornai, Kimmo Koskenniemi, Sandra Kübler, Roland Kuhn, Shankar Kumar, Hong-Kwang Kuo, Sadao Kurohashi;

Philippe Langlais, Guy Lapalme, Mirella Lapata, Eric Laporte, Staffan Larsson, Geunbae Lee, Lillian Lee, Oliver Lemon, Lori Levin, Roger Levy, Hang Li, Wei Li, Chin-Yew Lin, Ying Lin, Bing Liu, Yang Liu, Adam Lopez, Saturnino Luz, Caroline Lyon;

Bernardo Magnini, Milind Mahajan, Fran cois Mairesse, Suresh Manandhar, Lidia Mangu, Gideon Mann, Christopher Manning, Daniel Marcu, Mitchell Marcus, Katja Markert, David Martinez, Yuji Matsumoto, Takuya Matsuzaki, Irina Matveeva, John Maxwell, David McClosky, Ryan McDonald, Susan McRoy, Helen Meng, Wolfgang Menzel, Detmar Meurers, Rada Mihalcea, Eleni Miltsakaki, Gilad Mishne, Yusuke Miyao, Marie-Francine Moens, Saif Mohammad, Mehryar Mohri, Christof Monz, Robert Moore, Tatsunori Mori, Alessandro Moschitti, Karin Müller, Dragos Stefan Munteanu;

Vivi Nastase, Roberto Navigli, Mark-Jan Nederhof, Ani Nenkova, Hwee Tou Ng, Grace Ngai, Patrick Nguyen, Vincent Ng, Malvina Nissim, Cheng Niu, Joakim Nivre, Tadashi Nomoto, David Novick;

Tim Oates, Franz Och, Kemal Oflazer, Paul Ogilvie, Miles Osborne;

Sebastian Padó, Tim Paek, Chris Pal, David Palmer, Martha Palmer, Bo Pang, Patrick Pantel, Marius Pasca, Rebecca Passonneau, Slav Petrov, Fabio Pianesi, Paul Piwek, Ferran Pla, Massimo Poesio, Richard Power, David Powers, Sameer Pradhan, Rashmi Prasad, Adam Przepiórkowski, Stephen Pulman, Vasin Punyakanok, Matthew Purver;

Yan Qu, Chris Quirk;

Owen Rambow, Deepak Ravichandran, Giuseppe Riccardi, Sebastian Riedel, Stefan Riezler, German Rigau, Ellen Riloff, Sophie Rosset;

Kenji Sagae, Magnus Sahlgren, Helmut Schmid, Patrick Schone, Sabine Schulte im Walde, Tanja Schultz, Dale Schuurmans, Holger Schwenk, Frédérique Segond, Satoshi Sekine, Izhak Shafran, Libin Shen, Khalil Sima'an, Michel Simard, Wojciech Skut, David A. Smith, Noah A. Smith, Rion Snow, Stephen Soderland, Swapna Somasundaran, Radu Soricut, Caroline Sporleder, Richard Sproat, Padmini Srinivasan, Brad Starkie, Manfred Stede, Mark Steedman, Mark Stevenson, Matthew Stone, Veselin Stoyanov, Carlo Strapparava, Michael Strube, Jian Su, Maosong Sun, Mihai Surdeanu, Charles Sutton, Hisami Suzuki, Jun Suzuki, Marc Swerts, Stan Szpakowicz;

Maite Taboada, Hiroya Takamura, Isabelle Tellier, Joel Tetreault, Simone Teufel, Mariët Theune, Franck Thollard, Christoph Tillmann, Kristina Toutanova, Vivian Tsang, Jun'ichi Tsujii, Dan Tufis;

Nicola Ueffing, Ozlem Uzuner;

Antal van den Bosch, Ashish Venugopal, Renata Vieira, Marc Vilain, Begoña Villada Moirón, Aline Villavicencio, Carl Vogel, Stephan Vogel, Piek Vossen;

Qin Wang, Shaojun Wang, Wei Wang, Wen Wang, Ye-Yi Wang, Wayne Ward, Taro Watanabe, Andy Way, Nick Webb, Bonnie Webber, Ralph Weischedel, Edward Whittaker, Richard Wicentowski, Yorick Wilks, Theresa Wilson, Shuly Wintner, Dekai Wu, Xiaoyun Wu;

Aris Xanthos, Fei Xia, Eric Xing, Feiyu Xu, Peng Xu;

Scott Wen-tau Yih;

Fabio Zanzotto, Richard Zens, ChengXiang Zhai, Hao Zhang, Tong Zhang, Ying Zhang, Shubin Zhao, Guoding Zhou, Liang Zhou, Jerry Zhu, Michael Zock, Jelle Zuidema

CoNLL Shared Task Reviewers:

Jason Baldridge, Sabine Buchholz, James Clarke, Gülşen Eryiğit, Kilian Evang, Johan Hall, Julia Hockenmaier, Yuval Krymolowski, Sandra Kübler, Erwin Marsi, Ryan McDonald, Beáta Megyesi, Jens Nilsson, Joakim Nivre, Sebastian Riedel, Yannick Versley, Alexander Yeh, Deniz Yuret

Table of Contents

Conference Program Overview

Thursday, June 28, 2007

9:00–10:45	Session 1: Plenary Session
10:45–11:15	Morning Break
11:15–12:30	Sessions 2a and 2b
12:30–14:00	Lunch
14:00–15:40	Sessions 3a and 3b
15:40–16:00	Afternoon Break
16:00–18:30	Session 4: All Posters

Friday, June 29, 2007

9:00–10:40	Sessions 5a and 5b
10:40–11:15	Morning Break
11:15–12:30	Sessions 6a and 6b
12:30–14:00	Lunch
14:00–15:40	Sessions 7a and 7b
15:40–16:00	Afternoon Break
16:00–18:30	Session 8: All Posters

Saturday, June 30, 2007

9:00–10:00	Session 9: Plenary Session
10:00–10:50	Sessions 10a, 10b, and 10c
10:50–11:15	Morning Break
11:15–12:30	Sessions 11a, 11b, and 11c
12:30–14:00	Lunch
14:00–15:40	Sessions 12a, 12b, and 12c
15:40–16:15	Afternoon Break
16:15–17:30	Sessions 13a, 13b, and 13c
17:30	Concluding Session

Conference Program

Thursday, June 28, 2007

Session 1: Plenary Session

9:00–9:10 Opening Remarks

9:10–10:10 Invited Talk: *Baby Bayesians? Evidence for Statistical Hypothesis Selection in Infant Language Learning*
LouAnn Gerken, University of Arizona

10:15–10:45 *Modelling Compression with Discourse Constraints*
James Clarke and Mirella Lapata

Session 2a: Question Answering

11:15–11:40 *Using Semantic Roles to Improve Question Answering*
Dan Shen and Mirella Lapata

11:40–12:05 *What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA*
Mengqiu Wang, Noah A. Smith and Teruko Mitamura

12:05–12:30 *Learning Unsupervised SVM Classifier for Answer Selection in Web Question Answering*
Youzheng Wu, Ruiqiang Zhang, Xinhui Hu and Hideki Kashioka

Session 2b: Machine Translation

11:15–11:40 *Improving Word Alignment with Bridge Languages*
Shankar Kumar, Franz J. Och and Wolfgang Macherey

11:40–12:05 *Getting the Structure Right for Word Alignment: LEAF*
Alexander Fraser and Daniel Marcu

12:05–12:30 *Improving Statistical Machine Translation Using Word Sense Disambiguation*
Marine Carpuat and Dekai Wu

Thursday, June 28, 2007 (continued)

Session 3a: Generation, Summarization, and Discourse

14:00–14:25 *Large Margin Synchronous Generation and its Application to Sentence Compression*
Trevor Cohn and Mirella Lapata

14:25–14:50 *Incremental Text Structuring with Online Hierarchical Ranking*
Erdong Chen, Benjamin Snyder and Regina Barzilay

14:50–15:15 *Automatically Identifying the Arguments of Discourse Connectives*
Ben Wellner and James Pustejovsky

15:15–15:40 *Incremental Generation of Plural Descriptions: Similarity and Partitioning*
Albert Gatt and Kees van Deemter

Session 3b: Parsing

14:00–14:25 *A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors*
Joachim Wagner, Jennifer Foster and Josef van Genabith

14:25–14:50 *Characterizing the Errors of Data-Driven Dependency Parsing Models*
Ryan McDonald and Joakim Nivre

14:50–15:15 *Probabilistic Models of Nonprojective Dependency Trees*
David A. Smith and Noah A. Smith

15:15–15:40 *Structured Prediction Models via the Matrix-Tree Theorem*
Terry Koo, Amir Globerson, Xavier Carreras and Michael Collins

Thursday, June 28, 2007 (continued)

Session 4: All Posters (16:00–18:30)

Using Foreign Inclusion Detection to Improve Parsing Performance
Beatrice Alex, Amit Dubey and Frank Keller

LEDIR: An Unsupervised Algorithm for Learning Directionality of Inference Rules
Rahul Bhagat, Patrick Pantel and Eduard Hovy

Modelling Polysemy in Adjective Classes by Multi-Label Classification
Gemma Boleda, Sabine Schulte im Walde and Toni Badia

Improving Query Spelling Correction Using Web Search Results
Qing Chen, Mu Li and Ming Zhou

Towards Robust Unsupervised Personal Name Disambiguation
Ying Chen and James Martin

Compressing Trigram Language Models With Golomb Coding
Kenneth Church, Ted Hart and Jianfeng Gao

Joint Morphological and Syntactic Disambiguation
Shay B. Cohen and Noah A. Smith

Unsupervised Part-of-Speech Acquisition for Resource-Scarce Languages
Sajib Dasgupta and Vincent Ng

Semi-Supervised Classification for Extracting Protein Interaction Sentences using Dependency Parsing
Gunes Erkan, Arzucan Ozgur and Dragomir R. Radev

A Sequence Alignment Model Based on the Averaged Perceptron
Dayne Freitag and Shahram Khadivi

Instance Based Lexical Entailment for Ontology Population
Claudio Giuliano and Alfio Gliozzo

Recovering Non-Local Dependencies for Chinese
Yuqing Guo, Haifeng Wang and Josef van Genabith

Thursday, June 28, 2007 (continued)

Session 4: All Posters (16:00–18:30) (continued)

Exploiting Multi-Word Units in History-Based Probabilistic Generation
Deirdre Hogan, Conor Cafferkey, Aoife Cahill and Josef van Genabith

Hierarchical System Combination for Machine Translation
Fei Huang and Kishore Papineni

Using RBMT Systems to Produce Bilingual Corpus for SMT
Xiaoguang Hu, Haifeng Wang and Hua Wu

Why Doesn't EM Find Good HMM POS-Taggers?
Mark Johnson

Probabilistic Coordination Disambiguation in a Fully-Lexicalized Japanese Parser
Daisuke Kawahara and Sadao Kurohashi

A New Perceptron Algorithm for Sequence Labeling with Non-Local Features
Jun'ichi Kazama and Kentaro Torisawa

Extending a Thesaurus in the Pan-Chinese Context
Oi Yee Kwong and Benjamin K. Tsou

Low-Quality Product Review Detection in Opinion Summarization
Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang and Ming Zhou

Improving Statistical Machine Translation Performance by Training Data Selection and Optimization
Yajuan Lu, Jin Huang and Qun Liu

Topic Segmentation with Hybrid Document Indexing
Irina Matveeva and Gina-Anne Levow

Syntactic Re-Alignment Models for Machine Translation
Jonathan May and Kevin Knight

Detecting Compositionality of Verb-Object Combinations using Selectional Preferences
Diana McCarthy, Sriram Venkatapathy and Aravind Joshi

Explorations in Automatic Book Summarization
Rada Mihalcea and Hakan Ceylan

Thursday, June 28, 2007 (continued)

Session 4: All Posters (16:00–18:30) (continued)

Thursday, June 28, 2007 (continued)

Session 4: All Posters (16:00–18:30) (continued)

Friday, June 29, 2007

Session 5a: Semantics

9:00–9:25 *Generating Lexical Analogies Using Dependency Relations*
Andy Chiu, Pascal Poupart and Chrysanne DiMarco

9:25–9:50 *Cross-Lingual Distributional Profiles of Concepts for Measuring Semantic Distance*
Saif Mohammad, Iryna Gurevych, Graeme Hirst and Torsten Zesch

9:50–10:15 *Lexical Semantic Relatedness with Random Graph Walks*
Thad Hughes and Daniel Ramage

10:15–10:40 *Experimental Evaluation of LTAG-Based Features for Semantic Role Labeling*
Yudong Liu and Anoop Sarkar

Session 5b: Parsing

9:00–9:25 *Japanese Dependency Analysis Using the Ancestor-Descendant Relation*
Akihiro Tamura, Hiroya Takamura and Manabu Okumura

9:25–9:50 *A Discriminative Learning Model for Coordinate Conjunctions*
Masashi Shimbo and Kazuo Hara

9:50–10:15 *Recovery of Empty Nodes in Parse Structures*
Denis Filimonov and Mary Harper

10:15–10:40 *Treebank Annotation Schemes and Parser Evaluation for German*
Ines Rehbein and Josef van Genabith

Friday, June 29, 2007 (continued)

Session 6a: Document Analysis

11:15–11:40 *Semi-Markov Models for Sequence Segmentation*
Qinfeng Shi, Yasemin Altun, Alex Smola and S.V.N. Vishwanathan

11:40–12:05 *A Graph-Based Approach to Named Entity Categorization in Wikipedia Using Conditional Random Fields*
Yotaro Watanabe, Masayuki Asahara and Yuji Matsumoto

12:05–12:30 *MavenRank: Identifying Influential Members of the US Senate Using Lexical Centrality*
Anthony Fader, Dragomir R. Radev, Michael H. Crespin, Burt L. Monroe, Kevin M. Quinn and Michael Colaresi

Session 6b: Grammar Learning

11:15–11:40 *Bootstrapping Feature-Rich Dependency Parsers with Entropic Priors*
David A. Smith and Jason Eisner

11:40–12:05 *Online Learning of Relaxed CCG Grammars for Parsing to Logical Form*
Luke Zettlemoyer and Michael Collins

12:05–12:30 *The Infinite PCFG Using Hierarchical Dirichlet Processes*
Percy Liang, Slav Petrov, Michael Jordan and Dan Klein

Friday, June 29, 2007 (continued)

Session 7a: Information Extraction

14:00–14:25 *Exploiting Wikipedia as External Knowledge for Named Entity Recognition*
Jun'ichi Kazama and Kentaro Torisawa

14:25–14:50 *Large-Scale Named Entity Disambiguation Based on Wikipedia Data*
Silviu Cucerzan

14:50–15:15 *Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions*
Siddharth Patwardhan and Ellen Riloff

15:15–15:40 *Tree Kernel-Based Relation Extraction with Context-Sensitive Structured Parse Tree Information*
GuoDong Zhou, Min Zhang, DongHong Ji and QiaoMing Zhu

Session 7b: Machine Translation

14:00–14:25 *Chinese Syntactic Reordering for Statistical Machine Translation*
Chao Wang, Michael Collins and Philipp Koehn

14:25–14:50 *Binarizing Syntax Trees to Improve Syntax-Based Machine Translation Accuracy*
Wei Wang, Kevin Knight and Daniel Marcu

14:50–15:15 *What Can Syntax-Based MT Learn from Phrase-Based MT?*
Steve DeNeefe, Kevin Knight, Wei Wang and Daniel Marcu

15:15–15:40 *Online Large-Margin Training for Statistical Machine Translation*
Taro Watanabe, Jun Suzuki, Hajime Tsukada and Hideki Isozaki

Session 8: All Posters (16:00–18:30)

Consult the list of poster titles under Session 4.

Saturday, June 30, 2007

Session 9: Plenary Session

9:00–10:00 Invited Talk: *Hashing, Sketching, and Other Approximate Algorithms for High-Dimensional Data*
Piotr Indyk, Massachusetts Institute of Technology

Session 10a: Machine Learning (supervised classifiers)

10:00–10:25 *Scalable Term Selection for Text Categorization*
Jingyang Li and Maosong Sun

10:25–10:50 *Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem*
Jingbo Zhu and Eduard Hovy

Session 10b: Machine Learning (sequential models)

10:00–10:25 *Semi-Supervised Structured Output Learning Based on a Hybrid Generative and Discriminative Approach*
Jun Suzuki, Akinori Fujino and Hideki Isozaki

10:25–10:50 *Finding Good Sequential Model Structures using Output Transformations*
Edward Loper

Session 10c: Information Retrieval

10:00–10:25 *A Statistical Language Modeling Approach to Lattice-Based Spoken Document Retrieval*
Tee Kiah Chia, Haizhou Li and Hwee Tou Ng

10:25–10:50 *Learning Noun Phrase Query Segmentation*
Shane Bergsma and Qin Iris Wang

Saturday, June 30, 2007 (continued)

Session 11a: Information Extraction

11:15–11:40 *Bootstrapping Information Extraction from Field Books*
Sander Canisius and Caroline Sporleder

11:40–12:05 *Extracting Data Records from Unstructured Biomedical Full Text*
Donghui Feng, Gully Burns and Eduard Hovy

12:05–12:30 *Multiple Alignment of Citation Sentences with Conditional Random Fields and Posterior Decoding*
Ariel Schwartz, Anna Divoli and Marti Hearst

Session 11b: Machine Translation

11:15–11:40 *Large Language Models in Machine Translation*
Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och and Jeffrey Dean

11:40–12:05 *Factored Translation Models*
Philipp Koehn and Hieu Hoang

12:05–12:30 *Translating Unknown Words by Analogical Learning*
Philippe Langlais and Alexandre Patry

Session 11c: Phonetics and Phonology

11:15–11:40 *A Probabilistic Approach to Diachronic Phonology*
Alexandre Bouchard, Percy Liang, Thomas Griffiths and Dan Klein

11:40–12:05 *Learning Structured Models for Phone Recognition*
Slav Petrov, Adam Pauls and Dan Klein

12:05–12:30 *Inducing Search Keys for Name Filtering*
L. Karl Branting

Saturday, June 30, 2007 (continued)

Session 12a: CoNLL Shared Task Session (dependency parsing)

14:00–14:15 *The CoNLL 2007 Shared Task on Dependency Parsing*
Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel and Deniz Yuret

14:15–14:30 *Single Malt or Blended? A Study in Multilingual Parser Optimization*
Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson and Markus Saers

14:30–14:45 *Probabilistic Parsing Action Models for Multi-Lingual Dependency Parsing*
Xiangyu Duan, Jun Zhao and Bo Xu

14:45–15:00 *Fast and Robust Multilingual Dependency Parsing with a Generative Latent Variable Model*
Ivan Titov and James Henderson

15:00–15:15 *Multilingual Dependency Parsing Using Global Features*
Tetsuji Nakagawa

15:15–15:30 *Experiments with a Higher-Order Projective Dependency Parser*
Xavier Carreras

15:30–15:45 *Log-Linear Models of Non-Projective Trees, k-best MST Parsing and Tree-Ranking*
Keith Hall, Jiri Havelka and David A. Smith

Saturday, June 30, 2007 (continued)

Session 12b: Machine Translation

14:00–14:25 *Improving Translation Quality by Discarding Most of the Phrasetable*
Howard Johnson, Joel Martin, George Foster and Roland Kuhn

14:25–14:50 *Hierarchical Phrase-Based Translation with Suffix Arrays*
Adam Lopez

14:50–15:15 *An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems*
Wolfgang Macherey and Franz J. Och

15:15–15:40 *Learning to Find English to Chinese Transliterations on the Web*
Jian-Cheng Wu and Jason S. Chang

Session 12c: Word Senses

14:00–14:25 *Learning to Merge Word Senses*
Rion Snow, Sushant Prakash, Daniel Jurafsky and Andrew Y. Ng

14:25–14:50 *Improving Word Sense Disambiguation Using Topic Features*
Junfu Cai, Wee Sun Lee and Yee Whye Teh

14:50–15:15 *A Topic Model for Word Sense Disambiguation*
Jordan Boyd-Graber, David Blei and Xiaojin Zhu

15:15–15:40 *Validation and Evaluation of Automatically Acquired Multiword Expressions for Grammar Engineering*
Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart and Carlos Ramisch

Saturday, June 30, 2007 (continued)

Session 13a: CoNLL Shared Task Session (dependency parsing)

16:15–16:30 *Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles*
Kenji Sagae and Jun'ichi Tsujii

16:30–16:45 *Frustratingly Hard Domain Adaptation for Dependency Parsing*
Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca and Fernando Pereira

16:45–17:15 Analysis: Sandra Kübler, Ryan McDonald

17:15–17:30 Discussion

Session 13b: Sentiment

16:15–16:40 *Crystal: Analyzing Predictive Opinions on the Web*
Soo-Min Kim and Eduard Hovy

16:40–17:05 *Extracting Aspect-Evaluation and Aspect-Of Relations in Opinion Mining*
Nozomi Kobayashi, Kentaro Inui and Yuji Matsumoto

17:05–17:30 *Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents*
Nobuhiro Kaji and Masaru Kitsuregawa

Session 13c: Tagging

16:15–16:40 *Determining Case in Arabic: Learning Complex Linguistic Behavior Requires Complex Linguistic Features*
Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick and Mitch Marcus

16:40–17:05 *Mandarin Part-of-Speech Tagging and Discriminative Reranking*
Zhongqiang Huang, Mary Harper and Wen Wang

17:05–17:30 *Building Domain-Specific Taggers without Annotated (Domain) Data*
John Miller, Manabu Torii and K. Vijay-Shanker

Concluding Session

17:30 Closing Remarks

Additional CoNLL Shared Task Papers (dependency parsing)

Modelling Compression with Discourse Constraints

James Clarke and **Mirella Lapata**
School of Informatics, University of Edinburgh
2 Bucclecuch Place, Edinburgh EH8 9LW, UK
jclarke@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Sentence compression holds promise for many applications ranging from summarisation to subtitle generation and subtitle generation. The task is typically performed on isolated sentences without taking the surrounding context into account, even though most applications would operate over entire documents. In this paper we present a discourse informed model which is capable of producing document compressions that are coherent and informative. Our model is inspired by theories of local coherence and formulated within the framework of Integer Linear Programming. Experimental results show significant improvements over a state-of-the-art discourse agnostic approach.

1 Introduction

The computational treatment of sentence compression has recently attracted much attention in the literature. The task can be viewed as producing a summary of a single sentence that retains the most important information and remains grammatically correct (Jing 2000). Sentence compression is commonly expressed as a word deletion problem: given an input sentence of words $W = w_1, w_2, \ldots, w_n$, the aim is to produce a compression by removing any subset of these words (Knight and Marcu 2002).

Sentence compression can potentially benefit many applications. For example, in summarisation, a compression mechanism could improve the conciseness of the generated summaries (Jing 2000; Lin 2003). Sentence compression could be also used to automatically generate subtitles for television programs; the transcripts cannot usually be used verbatim due to the rate of speech being too high (Vandeghinste and Pan 2004). Other applications include compressing text to be displayed on small screens (Corston-Oliver 2001) such as mobile phones or PDAs, and producing audio scanning devices for the blind (Grefenstette 1998).

Most work to date has focused on a rather simple formulation of sentence compression that does not allow any rewriting operations, besides word removal. Moreover, compression is performed on isolated sentences without taking into account their surrounding context. An advantage of this simple view is that it renders sentence compression amenable to a variety of learning paradigms ranging from instantiations of the noisy-channel model (Galley and McKeown 2007; Knight and Marcu 2002; Turner and Charniak 2005) to Integer Linear Programming (Clarke and Lapata 2006a) and large-margin online learning (McDonald 2006).

In this paper we take a closer look at one of the simplifications associated with the compression task, namely that sentence reduction can be realised in isolation without making use of discourse-level information. This is clearly not true — professional abstracters often rely on contextual cues while creating summaries (Endres-Niggemeyer 1998). Furthermore, determining what information is important in a sentence is influenced by a variety of contextual factors such as the discourse topic, whether the sentence introduces new entities or events that have not been mentioned before, and the reader's background knowledge.

The simplification is also at odds with most applications of sentence compression which aim to create a shorter document rather than a single sentence. The resulting document must not only be grammat-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1–11, Prague, June 2007. ©2007 Association for Computational Linguistics

ical but also coherent if it is to function as a replacement for the original. However, this cannot be guaranteed without knowing how the discourse progresses from sentence to sentence. To give a simple example, a contextually aware compression system could drop a word or phrase from the current sentence, simply because it is not mentioned anywhere else in the document and is therefore deemed unimportant. Or it could decide to retain it for the sake of topic continuity.

We are interested in creating a compression model that is appropriate for documents and sentences. To this end, we assess whether discourse-level information is helpful. Our analysis is informed by two popular models of discourse, Centering Theory (Grosz et al. 1995) and lexical chains (Morris and Hirst 1991). Both approaches model *local coherence* — the way adjacent sentences bind together to form a larger discourse. Our compression model is an extension of the integer programming formulation proposed by Clarke and Lapata (2006a). Their approach is conceptually simple: it consists of a scoring function coupled with a small number of syntactic and semantic constraints. Discourse-related information can be easily incorporated in the form of additional constraints. We employ our model to perform sentence compression throughout a whole document (by compressing sentences sequentially) and evaluate whether the resulting text is understandable and informative using a question-answering task. Our method yields significant improvements over a discourse agnostic state-of-the-art compression model (McDonald 2006).

2 Related Work

Sentence compression has been extensively studied across different modelling paradigms and has received both generative and discriminative formulations. Most generative approaches (Galley and McKeown 2007; Knight and Marcu 2002; Turner and Charniak 2005) are instantiations of the noisy-channel model, whereas discriminative formulations include decision-tree learning (Knight and Marcu 2002), maximum entropy (Riezler et al. 2003), support vector machines (Nguyen et al. 2004), and large-margin learning (McDonald 2006). These models are trained on a parallel corpus of long *source* sentences and their *target* compressions. Using a rich feature set derived from parse trees, the models learn either which constituents to delete or which words to place adjacently in the compression output. Relatively few approaches dispense with the parallel corpus and generate compressions in an unsupervised manner using either a scoring function (Clarke and Lapata 2006a; Hori and Furui 2004) or compression rules that are approximated from a non-parallel corpus such as the Penn Treebank (Turner and Charniak 2005).

Our work differs from previous approaches in two key respects. First, we present a compression model that is contextually aware; decisions on whether to remove or retain a word (or phrase) are informed by its discourse properties (e.g., whether it introduces a new topic, whether it is semantically related to the previous sentence). Second, we apply our compression model to entire documents rather than isolated sentences. This is more in the spirit of real-world applications where the goal is to generate a condensed and coherent text. Previous work on summarisation has also utilised discourse information (e.g., Barzilay and Elhadad 1997; Daumé III and Marcu 2002; Marcu 2000; Teufel and Moens 2002). However, its application to document compression is novel to our knowledge.

3 Discourse Representation

Obtaining an appropriate representation of discourse is the first step towards creating a compression model that exploits contextual information. In this work we focus on the role of local coherence as this is prerequisite for maintaining global coherence. Ideally, we would like our compressed document to maintain the discourse flow of the original. For this reason, we automatically annotate the source document with discourse-level information which is subsequently used to inform our compression procedure. We first describe our algorithms for obtaining discourse annotations and then present our compression model.

3.1 Centering Theory

Centering Theory (Grosz et al. 1995) is an entity-orientated theory of local coherence and salience. Although an utterance in discourse may contain several entities, it is assumed that a *single entity* is salient or "centered", thereby representing the current focus. One of the main claims underlying centering is that discourse segments in which successive utterances contain common centers are more coherent than segments where the center repeatedly changes.

Each utterance U_i in a discourse segment has a list of *forward-looking centers*, $C_f(U_i)$ and a *unique backward-looking center*, $C_b(U_i)$. $C_f(U_i)$ represents a ranking of the entities invoked by U_i according to their salience. The C_b of the current utterance U_i, is the highest-ranked element in $C_f(U_{i-1})$ that is also in U_i. The C_b thus links U_i to the previous discourse, but it does so *locally* since $C_b(U_i)$ is chosen from U_{i-1}.

Centering Algorithm So far we have presented centering without explicitly stating how the concepts "utterance", "entities" and "ranking" are instantiated. A great deal of research has been devoted into fleshing these out and many different instantiations have been developed in the literature (see Poesio et al. 2004 for details). Since our aim is to identify centers in discourse automatically, our parameter choice is driven by two considerations, robustness and ease of computation.

We therefore follow previous work (e.g., Miltsakaki and Kukich 2000) in assuming that the unit of an utterance is the sentence (i.e., a main clause with accompanying subordinate and adjunct clauses). This is in line with our compression task which also operates over sentences. We determine which entities are invoked by a sentence using two methods. First, we perform named entity identification and coreference resolution on each document using LingPipe[1], a publicly available system. Named entities and all remaining nouns are added to the C_f list. Entity matching between sentences is required to determine the C_b of a sentence. This is done using the named entity's unique identifier (as provided by LingPipe) or by the entity's surface form in the case of nouns not classified as named entities.

Entities are ranked according to their grammatical roles; subjects are ranked more highly than objects, which are in turn ranked higher than other grammatical roles (Grosz et al. 1995); ties are broken using left-to-right ordering of the grammatical roles in the sentence (Tetreault 2001). We identify grammatical roles with RASP (Briscoe and Carroll 2002). Formally, our centering algorithm is as follows (where U_i corresponds to sentence i):

1. Extract entities from U_i.
2. Create $C_f(U_i)$ by ranking the entities in U_i according to their grammatical role (subjects $>$ objects $>$ others).
3. Find the highest ranked entity in $C_f(U_{i-1})$ which occurs in $C_f(U_i)$, set the entity to be $C_b(U_i)$.

The above procedure involves several automatic steps (named entity recognition, coreference resolution, identification of grammatical roles) and will unavoidably produce some noisy annotations. So, there is no guarantee that the right C_b will be identified or that all sentences will be marked with a C_b. The latter situation also occurs in passages that contain abrupt changes in topic. In such cases, none of the entities realised in U_i will occur in $C_f(U_{i-1})$. Rather than accept that discourse information may be absent in a sentence, we turn to lexical chains as an alternative means of capturing topical content within a document.

3.2 Lexical Chains

Lexical cohesion refers to the degree of semantic relatedness observed among lexical items in a document. The term was coined by Halliday and Hasan (1976) who observed that coherent documents tend to have more related terms or phrases than incoherent ones. A number of linguistic devices can be used to signal cohesion; these range from repetition, to synonymy, hyponymy and meronymy. Lexical chains are a representation of lexical cohesion as sequences of semantically related words (Morris and Hirst 1991) and provide a useful means for describing the topic flow in discourse. For instance, a document with many different lexical chains will probably contain several topics. And main topics will tend to be represented by dense and long chains. Words participating in such chains are important for our compression task — they reveal what the document is about — and in all likelihood should not be deleted.

Lexical Chains Algorithm Barzilay and Elhadad (1997) describe a technique for text summarisation based on lexical chains. Their algorithm uses WordNet to build chains of nouns (and noun compounds). These are ranked heuristically by a score based on their length and homogeneity. A summary is then produced by extracting sentences corresponding to

[1] LingPipe can be downloaded from `http://www.alias-i.com/lingpipe/`.

strong chains, i.e., chains whose score is two standard deviations above the average score.

Like Barzilay and Elhadad (1997), we wish to determine which lexical chains indicate the most prevalent discourse topics. Our assumption is that terms belonging to these chains are indicative of the document's main focus and should therefore be retained in the compressed output. Barzilay and Elhadad's scoring function aims to identify sentences (for inclusion in a summary) that have a high concentration of chain members. In contrast, we are interested in chains that span several sentences. We thus score chains according to the number of sentences their terms occur in. For example, the chain $\{house_3, home_3, loft_3, house_5\}$ (where $word_i$ denotes *word* occurring in sentence *i*) would be given a score of two as the terms only occur in two sentences. We assume that a chain signals a prevalent discourse topic if it occurs throughout more sentences than the average chain. The scoring algorithm is outlined more formally below:

1. Compute the lexical chains for the document.
2. $Score(Chain) = Sentences(Chain)$.
3. Discard chains if $Score(Chain) < Avg(Score)$.
4. Mark terms from the remaining chains as being the focus of the document.

We use the method of Galley and McKeown (2003) to compute lexical chains for each document.[2] This is an improved version of Barzilay and Elhadad's (1997) original algorithm.

Before compression takes place, all documents are pre-processed using the centering and lexical chain algorithms described above. In each sentence we mark the center $C_b(U_i)$ if one exists. Words (or phrases) that are present in the current sentence and function as the center in the next sentence $C_b(U_{i+1})$ are also flagged. Finally, words are marked if they are part of a prevalent chain. An example of our discourse annotation is given in Figure 1.

Bad weather dashed hopes of attempts to halt the flow$_1$ during what was seen as a lull in the lava's momentum. Experts say that even if the eruption stopped today$_2$, the pressure of lava piled up behind for six miles$_3$ would bring debris cascading down on to the town anyway. Some estimate the volcano is pouring out one million tons of debris a day$_2$, at a rate$_1$ of 15ft$_3$ per second$_2$, from a fissure that opened in mid-December.
The Italian Army yesterday$_2$ detonated 400lb of dynamite 3,500 feet up Mount Etna's slopes.

Figure 1: Excerpt of document from our test set with discourse annotations. Centers are in double boxes; terms occurring in lexical chains are in oval boxes. Words with the same subscript are members of the same chain (e.g., *today*, *day*, *second*, *yesterday*)

4 The Compression Model

Our model is an extension of the approach put forward in Clarke and Lapata (2006a). Their work tackles sentence compression as an optimisation problem. Given a long sentence, a compression is formed by retaining the words that maximise a scoring function. The latter is essentially a language model coupled with a few constraints ensuring that the resulting output is grammatical. The language model and the constraints are encoded as linear inequalities whose solution is found using Integer Linear Programming (ILP, Vanderbei 2001; Winston and Venkataramanan 2003).

We selected this model for several reasons. First it does not require a parallel corpus and thus can be ported across domains and text genres, whilst delivering state-of-the-art results (see Clarke and Lapata 2006a for details). Second, discourse-level information can be easily incorporated by augmenting the constraint set. This is not the case for other approaches (e.g., those based on the noisy channel model) where compression is modelled by grammar rules indicating which constituents to delete in a syntactic context. Third, the ILP framework delivers a globally optimal solution by searching over the entire compression space[3] without employing heuristics or approximations during decoding.

We begin by recapping the formulation of Clarke and Lapata (2006a). Let $W = w_1, w_2, \ldots, w_n$ denote a sentence for which we wish to generate a compression. A set of binary decision variables represent whether each word w_i should be included in the

[2] The software is available from `http://www1.cs.columbia.edu/~galley/`.

[3] For a sentence of length *n*, there are 2^n compressions.

compression or not. Let:

$$y_i = \begin{cases} 1 & \text{if } w_i \text{ is in the compression} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1 \dots n]$$

A trigram language model forms the backbone of the compression model. The language model is formulated as an integer program with the introduction of extra decision variables indicating which *word sequences* should be retained or dropped from the compression. Let:

$$p_i = \begin{cases} 1 \text{ if } w_i \text{ starts the compression} \\ 0 \text{ otherwise} \end{cases} \forall i \in [1 \dots n]$$

$$q_{ij} = \begin{cases} 1 \text{ if sequence } w_i, w_j \text{ ends} \\ \quad \text{the compression} & \forall i \in [1 \dots n-1] \\ 0 \text{ otherwise} & \forall j \in [i+1 \dots n] \end{cases}$$

$$x_{ijk} = \begin{cases} 1 \text{ if sequence } w_i, w_j, w_k & \forall i \in [1 \dots n-2] \\ \quad \text{is in the compression} & \forall j \in [i+1 \dots n-1] \\ 0 \text{ otherwise} & \forall k \in [j+1 \dots n] \end{cases}$$

The objective function is expressed in Equation (1). It is the sum of all possible trigrams multiplied by the appropriate decision variable. The objective function also includes a significance score for each word multiplied by the decision variable for that word (see the last summation term in (1)). This score highlights important content words in a sentence and is defined in Section 4.1.

$$\begin{aligned} \max z \quad = \quad & \sum_{i=1}^{n} p_i \cdot P(w_i|\text{start}) \\ & + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} x_{ijk} \cdot P(w_k|w_i, w_j) \\ & + \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} q_{ij} \cdot P(\text{end}|w_i, w_j) \\ & + \sum_{i=1}^{n} y_i \cdot I(w_i) \qquad (1) \end{aligned}$$

subject to:

$$y_i, p_i, q_{ij}, x_{ijk} = 0 \text{ or } 1 \qquad (2)$$

A set of *sequential* constraints[4] are added to the problem to only allow results which combine valid trigrams.

4.1 Significance Score

The significance score is an attempt at capturing the gist of a sentence. It gives more weight to content words that appear in the deepest level of embedding in the syntactic tree representing the source sentence:

$$I(w_i) = \frac{l}{N} \cdot f_i \log \frac{F_a}{F_i} \qquad (3)$$

The score is computed over a large corpus where w_i is a content word (i.e., a noun or verb), f_i and F_i are the frequencies of w_i in the document and corpus respectively, and F_a is the sum of all content words in the corpus. l is the number of clause constituents above w_i, and N is the deepest level of embedding.

4.2 Sentential Constraints

The model also contains a small number of sentence-level constraints. Their aim is to preserve the meaning and structure of the original sentence as much as possible. The majority of constraints revolve around modification and argument structure and are defined over parse trees or grammatical relations. For example, the following constraint template disallows the inclusion of modifiers (e.g., nouns, adjectives) without their head words:

$$\begin{gathered} y_i - y_j \geq 0 \qquad (4) \\ \forall i, j : w_j \text{ modifies } w_i \end{gathered}$$

Other constraints force the presence of modifiers when the head is retained in the compression. This way, it is ensured that negation will be preserved in the compressed output:

$$\begin{gathered} y_i - y_j = 0 \qquad (5) \\ \forall i, j : w_j \text{ modifies } w_i \ \wedge \ w_j = \text{not} \end{gathered}$$

Argument structure constraints make sure that the resulting compression has a canonical argument structure. For instance a constraint ensures that if a verb is present in the compression then so are its arguments:

$$\begin{gathered} y_i - y_j = 0 \qquad (6) \\ \forall i, j : w_j \in \text{subject/object of verb } w_i \end{gathered}$$

Finally, Clarke and Lapata (2006a) propose one discourse constraint which forces the system to preserve personal pronouns in the compressed output:

$$\begin{gathered} y_i = 1 \qquad (7) \\ \forall i : w_i \in \text{ personal pronouns} \end{gathered}$$

[4] We have omitted sequential constraints due to space limitations. The full details are given in Clarke and Lapata (2006a).

4.3 Discourse Constraints

In addition to the constraints described above, our model includes constraints relating to the centering and lexical chains representations discussed in Section 3. Recall that after some pre-processing, each sentence is marked with: its own center $C_b(U_i)$, the center $C_b(U_{i+1})$ of the sentence following it and words that are members of high scoring chains corresponding to the document's focus. We introduce two new types of constraints based on these additional knowledge sources.

The first constraint is the centering constraint which operates over adjacent sentences. It ensures that the C_b identified in the source sentence is retained in the target compression. If present, the entity realised as the C_b in the following sentence is also retained:

$$y_i = 1 \qquad (8)$$
$$\forall i : w_i \in \{C_b(U_i), C_b(U_{i+1})\}$$

Consider for example the discourse in Figure 1. The constraints generated from Equation (8) will require the compression to retain *lava* in the first two sentences and *debris* in sentences two and three.

We also add a lexical chain constraint. This applies only to nouns which are members of prevalent chains:

$$y_i = 1 \qquad (9)$$
$$\forall i : w_i \in \text{document focus lexical chain}$$

This constraint is complementary to the centering constraint; the sentences it applies to do not have to be adjacent and the entities under consideration are not restricted to a specific syntactic role (e.g., subject or object). See for instance the words *flow* and *rate* in Figure 1 which are members of the same chain (marked with subscript one). According to constraint (9) both words must be included in the compressed document.

The constraints just described ensure that the compressed document will retain the discourse flow of the original and will preserve terms indicative of important topics. We argue that these constraints will additionally benefit sentence-level compression, as words which are not signalled as discourse relevant can be dropped.

4.4 Applying the Constraints

Our compression system is given a (sentence separated) document as input. The ILP model just presented is then applied sequentially to all sentences to generate a compressed version of the original. We thus create and solve an ILP for every sentence.[5] In the formulation of Clarke and Lapata (2006a) a significance score (see Section 4.1) highlights which nouns and verbs to include in the compression. As far as nouns are concerned, our discourse constraints perform a similar task. Thus, when a sentence contains discourse annotations, we are inclined to trust them more and only calculate the significance score for verbs.

During development it was observed that applying all discourse constraints simultaneously (see Equations (7)–(9)) results in relatively long compressions. To counter this, we employ these constraints using a back-off strategy that relies on progressively less reliable information. Our back-off model works as follows: if centering information is present, we apply the appropriate constraints (Equation (8)). If no centers are present, we back-off to the lexical chain information using Equation (9), and in the absence of the latter we back-off to the pronoun constraint (Equation (7)). Finally, if discourse information is entirely absent from the sentence, we default to the significance score. Sentential constraints (see Section 4.2) are applied throughout irrespectively of discourse constraints. In our test data (see Section 5 for details), the centering constraint was used in 68.6% of the sentences. The model backed off to lexical chains for 13.7% of the test sentences, whereas the pronoun constraint was applied in 8.5%. Finally, the noun and verb significance score was used on the remaining 9.2%. An example of our system's output for the text in Figure 1 is given in Figure 2.

5 Experimental Set-up

In this section we present our experimental set-up. We briefly introduce the model used for comparison with our approach and give details regarding our compression corpus and parameter estimation. Finally, we describe our evaluation methodology.

[5] We use the publicly available *lp_solve* solver (`http://www.geocities.com/lpsolve/`).

> Bad weather dashed hopes to halt the flow during what was seen as lull in lava's momentum. Experts say that even if eruption stopped, the pressure of lava piled would bring debris cascading. Some estimate volcano is pouring million tons of debris from fissure opened in mid-December. The Army yesterday detonated 400lb of dynamite.

Figure 2: System output on excerpt from Figure 1.

Comparison with state-of-the-art An obvious evaluation experiment would involve comparing the ILP model without any discourse constraints against the discourse informed model presented in this work. Unfortunately, the two models obtain markedly different compression rates[6] which renders the comparison of their outputs problematic. To put the comparison on an equal footing, we evaluated our approach against a state-of-the-art model that achieves a compression rate similar to ours without taking discourse-level information into account. McDonald (2006) formalises sentence compression in a discriminative large-margin learning framework as a classification task: pairs of words from the source sentence are classified as being adjacent or not in the target compression. A large number of features are defined over words, parts of speech, phrase structure trees and dependencies. These are gathered over adjacent words in the compression and the words in-between which were dropped.

It is important to note that McDonald (2006) is not a straw-man system. It achieves highly competitive performance compared with Knight and Marcu's (2002) noisy channel and decision tree models. Due to its discriminative nature, the model is able to use a large feature set and to optimise compression accuracy directly. In other words, McDonald's model has a head start against our own model which does not utilise a parallel corpus and has only a few constraints. The comparison of the two systems allows us to investigate whether discourse information is redundant when using a powerful sentence compression model.

Corpus Previous work on sentence compression has used almost exclusively the Ziff-Davis, a compression corpus derived automatically from document-abstract pairs (Knight and Marcu 2002). Unfortunately, this corpus is not suitable for our purposes since it consists of isolated sentences. We thus created a document-based compression corpus manually. Following Clarke and Lapata (2006a), we asked annotators to produce compressions for 82 stories (1,629 sentences) from the BNC and the LA Times Washington Post.[7] 48 documents (962 sentences) were used for training, 3 for development (63 sentences), and 31 for testing (604 sentences).

Parameter Estimation Our parameters for the ILP model followed closely Clarke and Lapata (2006a). We used a language model trained on 25 million tokens from the North American News corpus. The significance score was based on 25 million tokens from the same corpus. Our re-implementation of McDonald (2006) used an identical feature set, and a slightly modified loss function to encourage compression on our data set.[8]

Evaluation Previous studies evaluate how well-formed the automatically derived compressions are out of context. The target sentences are typically rated by naive subjects on two dimensions, grammaticality and importance (Knight and Marcu 2002). Automatic evaluation measures have also been proposed. Riezler et al. (2003) compare the grammatical relations found in the system output against those found in a gold standard using F-score which Clarke and Lapata (2006b) show correlates reliably with human judgements.

Following previous work, sentence-based compressions were evaluated automatically using F-score computed over grammatical relations which we obtained by RASP (Briscoe and Carroll 2002). Besides individual sentences, our goal was to evaluate the compressed document as whole. Our evaluation methodology was motivated by two questions: (1) are the documents readable? and (2) how much key information is preserved between the source document and its target compression? We assume here that the compressed document is to function as a replacement for the original. We can thus measure the extent to which the compressed version can be

[6] The discourse agnostic ILP model has a compression rate of 81.2%; when discourse constraints are include the rate drops to 65.4%.

[7] The corpus is available from `http://homepages.inf.ed.ac.uk/s0460084/data/`.

[8] McDonald's (2006) results are reported on the Ziff-Davis corpus.

What is posing a threat to the town? (lava)
What hindered attempts to stop the lava flow? (bad weather)
What did the Army do first to stop the lava flow? (detonate explosives)

Figure 3: Example questions with answer key.

used to find answers for questions which are derived from the original and represent its core content.

We therefore employed a question-answering evaluation paradigm which has been previously used for summarisation evaluation and text comprehension (Mani et al. 2002; Morris et al. 1992). The overall objective of our Q&A task is to determine how accurate each document (generated by different compression systems) is at answering questions. For this we require a methodology for constructing Q&A pairs and for scoring each document.

Two annotators were independently instructed to create Q&A pairs for the original documents in the test set. Each annotator read the document and then drafted no more than ten questions and answers related to its content. Annotators were asked to create factual-based questions which required an unambiguous answer; these were typically who/what/where/when/how style questions. Annotators then compared and revised their question-answer pairs to create a common agreed upon set. Revisions typically involved merging questions, rewording and simplifying questions, and in some cases splitting a question into multiple questions. Documents for which too few questions were created or for which questions or answers were too ambiguous were removed. This left an evaluation set of six documents with between five to eight concise questions per document. Some example questions corresponding to the document from Figure 1 are given in Figure 3; correct answers are shown in parentheses.

Compressed documents and their accompanying questions were presented to human subjects who were asked to provide answers as best they could. We elicited answers for six documents in three compression conditions: gold standard, using the ILP discourse model, and McDonald's (2006) model. Each participant was also asked to rate the readability of the compressed document on a seven point scale. A Latin Square design prevented participants from seeing two different compressions of the same document.

Model	CompR	F-Score
McDonald	60.1%	36.0%*
Discourse ILP	65.4%	39.6%
Gold Standard	70.3%	—–

Table 1: Compression results: compression rate and relation-based F-score; * sig. diff. from Discourse ILP ($p < 0.05$ using the Student t test).

Model	Readability	Q&A
McDonald	2.6*	53.7%*†
Discourse ILP	3.0*	68.3%
Gold Standard	5.5†	80.7%

Table 2: Human Evaluation Results: average readability ratings and average percentage of questions answered correctly. *: sig. diff. from Gold Standard; †: sig. diff. from Discourse ILP.

The study was conducted remotely over the Internet. Participants were presented with a set of instructions that explained the Q&A task and provided examples. Subjects were first asked to read the compressed document and rate its readability. Questions were then presented one at a time and participants were allowed to consult the document for the answer. Once a participant had provided an answer they were not allowed to modify it. Thirty unpaid volunteers took part in our Q&A study. All were self reported native English speakers.

The answers provided by the participants were scored against the answer key. Answers were considered correct if they were identical to the answer key or subsumed by it. For instance, *Mount Etna* was considered a right answer to the first question from Figure 3. A compressed document receives a full score if subjects have answered all questions relating to it correctly.

6 Results

As a sanity check, we first assessed the compressions produced by our model and McDonald (2006) on a sentence-by-sentence basis without taking the documents into account. There is no hope for generating shorter documents if the compressed sentences are either too wordy or too ungrammatical. Table 1 shows the compression rates (CompR) for the two systems and evaluates the quality of their output using F-score based on grammatical relations. As can be seen, the Discourse ILP compressions are slightly longer than McDonald (65.4% vs. 60.1%) but closer to the human gold standard (70.3%). This is not surprising, the Discourse ILP model takes the entire document into account, and compression decisions will be slightly more conservative. The Discourse ILP's output is significantly better than McDonald in terms of F-score, indicating that discourse-level information is generally helpful. Both systems could use further improvement as inter-annotator agreement on this data yields an F-score of 65.8%.

Let us now consider the results of our document-based evaluation. Table 2 shows the mean readability ratings obtained for each system and the percentage of questions answered correctly. We used an Analysis of Variance (ANOVA) to examine the effect of compression type (McDonald, Discourse ILP, Gold Standard). The ANOVA revealed a reliable effect on both readability and Q&A. Post-hoc Tukey tests showed that McDonald and the Discourse ILP model do not differ significantly in terms of readability. However, they are significantly less readable than the gold standard ($\alpha < 0.05$). For the Q&A task we observe that our system is significantly better than McDonald ($\alpha < 0.05$) and not significantly worse than the gold standard.

These results indicate that the automatic systems lag behind the human gold standard in terms of readability. When reading entire documents, subjects are less tolerant of ungrammatical constructions. We also find out that despite relatively low readability, the documents are overall understandable. The discourse informed model generates more informative documents — the number of questions answered correctly increases by 15% in comparison to McDonald. This is an encouraging result suggesting that there may be advantages in developing compression models that exploit contextual information.

7 Conclusions and Future Work

In this paper we proposed a novel method for automatic sentence compression. Central in our approach is the use of discourse-level information which we argue is an important prerequisite for document (as opposed to sentence) compression. Our model uses integer programming for inferring globally optimal compressions in the presence of linguistically motivated constraints. Our discourse constraints aim to capture local coherence and are inspired by centering theory and lexical chains. We showed that our model can be successfully employed to produce compressed documents that preserve most of the original's core content.

Our approach to document compression differs from most summarisation work in that our summaries are fairly long. However, we believe this is the first step into understanding how compression can help summarisation. In the future, we will interface our compression model with sentence extraction. The discourse annotations can help guide the extraction method into selecting topically related sentences which can consequently be compressed together. The compression rate can be tailored through additional constraints which act on the output length to ensure precise word limits are obeyed.

We also plan to study the effect of global discourse structure (Daumé III and Marcu 2002) on the compression task. In general, we will assess the impact of discourse information more systematically by incorporating it into generative and discriminative modelling paradigms.

Acknowledgements We are grateful to Ryan McDonald for his help with the re-implementation of his system and our annotators Vasilis Karaiskos and Sarah Luger. Thanks to Simone Teufel, Alex Lascarides, Sebastian Riedel, and Bonnie Webber for insightful comments and suggestions. Lapata acknowledges the support of EPSRC (grant GR/T04540/01).

References

Barzilay, R. and M. Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS), ACL-97*.

Briscoe, E. J. and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC–2002)*. Las Palmas, Gran Canaria, pages 1499–1504.

Clarke, James and Mirella Lapata. 2006a. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the COLING/ACL 2006 Main Conference*

Poster Sessions. Sydney, Australia, pages 144–151.

Clarke, James and Mirella Lapata. 2006b. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 377–384.

Corston-Oliver, Simon. 2001. Text Compaction for Display on Very Small Screens. In *Proceedings of the NAACL Workshop on Automatic Summarization*. Pittsburgh, PA, pages 89–98.

Daumé III, Hal and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*. Philadelphia, PA, pages 449–456.

Endres-Niggemeyer, Brigitte. 1998. *Summarising Information*. Springer, Berlin.

Galley, Michel and Kathleen McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI–03)*. pages 1486–1488.

Galley, Michel and Kathleen McKeown. 2007. Lexicalized markov grammars for sentence compression. In *In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT–2007)*. Rochester, NY.

Grefenstette, Gregory. 1998. Producing Intelligent Telegraphic Text Reduction to Provide an Audio Scanning Service for the Blind. In *Proceedings of the AAAI Symposium on Intelligent Text Summarization*. Stanford, CA, pages 111–117.

Grosz, Barbara J., Scott Weinstein, and Aravind K. Joshi. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2):203–225.

Halliday, M. A. K. and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman, London.

Hori, Chiori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems* E87-D(1):15–25.

Jing, Hongyan. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the 6th conference on Applied Natural Language Processing (ANLP–2000)*. Seattle, WA, pages 310–315.

Knight, Kevin and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence* 139(1):91–107.

Lin, Chin-Yew. 2003. Improving summarization performance by sentence compression — a pilot study. In *Proceedings of the 6th International Workshop on Information Retrieval with Asian Languages*. Sapporo, Japan, pages 1–8.

Mani, Inderjeet, Gary Klein, David House, Lynette Hirschman, Therese Firmin, and Beth Sundheim. 2002. SUMMAC: A text summarization evaluation. *Natural Language Engineering* 8(1):43–68.

Marcu, Daniel. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press, Cambridge, MA.

McDonald, Ryan. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proceedings of the 11th EACL*. Trento, Italy.

Miltsakaki, Eleni and Karen Kukich. 2000. The role of centering theory's rough-shift in the teaching and evaluation of writing skills. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL–2000)*. pages 408–415.

Morris, A., G. Kasper, and D. Adams. 1992. The effects and limitations of automated text condensing on reading comprehension performance. *Information Systems Research* 3(1):17–35.

Morris, Jane and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics* 17(1):21–48.

Nguyen, Minh Le, Akira Shimazu, Susumu Horiguchi, Tu Bao Ho, and Masaru Fukushi. 2004. Probabilistic sentence reduction using support vector machines. In *Proceedings of the 20th COLING*. Geneva, Switzerland, pages 743–749.

Poesio, Massimo, Rosemary Stevenson, Barbara Di Eugenio, and Janet Hitzeman. 2004. Centering: a parametric theory and its instantiations. *Computational Linguistics* 30(3):309–363.

Riezler, Stefan, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the HLT/NAACL*. Edmonton, Canada, pages 118–125.

Tetreault, Joel R. 2001. A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics* 27(4):507–520.

Teufel, Simone and Marc Moens. 2002. Summarizing scientific articles – experiments with relevance and rhetorical status. *Computational Linguistics* 28(4):409–446.

Turner, Jenine and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd ACL*. Ann Arbor, MI, pages 290–297.

Vandeghinste, Vincent and Yi Pan. 2004. Sentence compression for automated subtitling: A hybrid approach. In *Proceedings of the ACL Workshop on Text Summarization*. Barcelona, Spain, pages 89–95.

Vanderbei, Robert J. 2001. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, Boston, 2nd edition.

Winston, Wayne L. and Munirpallam Venkataramanan. 2003. *Introduction to Mathematical Programming*. Brooks/Cole.

Using Semantic Roles to Improve Question Answering

Dan Shen
Spoken Language Systems
Saarland University
Saarbruecken, Germany
dan@lsv.uni-saarland.de

Mirella Lapata
School of Informatics
University of Edinburgh
Edinburgh, UK
mlap@inf.ed.ac.uk

Abstract

Shallow semantic parsing, the automatic identification and labeling of sentential constituents, has recently received much attention. Our work examines whether semantic role information is beneficial to question answering. We introduce a general framework for answer extraction which exploits semantic role annotations in the FrameNet paradigm. We view semantic role assignment as an optimization problem in a bipartite graph and answer extraction as an instance of graph matching. Experimental results on the TREC datasets demonstrate improvements over state-of-the-art models.

1 Introduction

Recent years have witnessed significant progress in developing methods for the automatic identification and labeling of semantic roles conveyed by sentential constituents.[1] The success of these methods, often referred to collectively as *shallow semantic parsing* (Gildea and Jurafsky, 2002), is largely due to the availability of resources like FrameNet (Fillmore et al., 2003) and PropBank (Palmer et al., 2005), which document the surface realization of semantic roles in real world corpora.

More concretely, in the FrameNet paradigm, the meaning of predicates (usually verbs, nouns, or adjectives) is conveyed by *frames*, schematic representations of situations. Semantic roles (or *frame elements*) are defined for each frame and correspond to salient entities present in the evoked situation. Predicates with similar semantics instantiate the same frame and are attested with the same roles. The FrameNet database lists the surface syntactic realizations of semantic roles, and provides annotated example sentences from the British National Corpus. For example, the frame *Commerce_Sell* has three core semantic roles, namely *Buyer*, *Goods*, and *Seller* — each expressed by an indirect object, a direct object, and a subject (see sentences (1a)–(1c)). It can also be attested with non-core (peripheral) roles (e.g., *Means, Manner*, see (1d) and (1e)) that are more generic and can be instantiated in several frames, besides *Commerce_Sell*. The verbs *sell*, *vend*, and *retail* can evoke this frame, but also the nouns *sale* and *vendor*.

(1)
a. [Lee]$_{Seller}$ sold a textbook [to Abby]$_{Buyer}$.
b. [Kim]$_{Seller}$ sold [the sweater]$_{Goods}$.
c. [My company]$_{Seller}$ has sold [more than three million copies]$_{Goods}$.
d. [Abby]$_{Seller}$ sold [the car]$_{Goods}$ [for cash]$_{Means}$.
e. [He]$_{Seller}$ [reluctanctly]$_{Manner}$ sold [his rock]$_{Goods}$.

By abstracting over surface syntactic configurations, semantic roles offer an important first step towards deeper text understanding and hold promise for a range of applications requiring broad coverage semantic processing. Question answering (QA) is often cited as an obvious beneficiary of semantic

[1] The approaches are too numerous to list; we refer the interested reader to Carreras and Màrquez (2005) for an overview.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 12–21, Prague, June 2007. ©2007 Association for Computational Linguistics

role labeling (Gildea and Jurafsky, 2002; Palmer et al., 2005; Narayanan and Harabagiu, 2004). Faced with the question *Q: What year did the U.S. buy Alaska?* and the retrieved sentence *S: . . . before Russia sold Alaska to the United States in 1867*, a hypothetical QA system must identify that *United States* is the *Buyer* despite the fact that it is attested in one instance as a subject and in another as an object. Once this information is known, isolating the correct answer (i.e., *1867*) can be relatively straightforward.

Although conventional wisdom has it that semantic role labeling ought to improve answer extraction, surprising little work has been done to this effect (see Section 2 for details) and initial results have been mostly inconclusive or negative (Sun et al., 2005; Kaisser, 2006). There are at least two good reasons for these findings. First, shallow semantic parsers trained on declarative sentences will typically have poor performance on questions and generally on out-of-domain data. Second, existing resources do not have exhaustive coverage and recall will be compromised, especially if the question answering system is expected to retrieve answers from unrestricted text. Since FrameNet is still under development, its coverage tends to be more of a problem in comparison to other semantic role resources such as PropBank.

In this paper we propose an answer extraction model which effectively incorporates FrameNet-style semantic role information. We present an automatic method for semantic role assignment which is conceptually simple and does not require extensive feature engineering. A key feature of our approach is the comparison of dependency relation paths attested in the FrameNet annotations and raw text. We formalize the search for an optimal role assignment as an optimization problem in a bipartite graph. This formalization allows us to find an exact, globally optimal solution. The graph-theoretic framework goes some way towards addressing coverage problems related with FrameNet and allows us to formulate answer extraction as a graph matching problem. As a byproduct of our main investigation we also examine the issue of FrameNet coverage and show how much it impacts performance in a TREC-style question answering setting.

In the following section we provide an overview of existing work on question answering systems that exploit semantic role-based lexical resources. Then we define our learning task and introduce our approach to semantic role assignment and answer extraction in the context of QA. Next, we present our experimental framework and data. We conclude the paper by presenting and discussing our results.

2 Related Work

Question answering systems have traditionally depended on a variety of lexical resources to bridge surface differences between questions and potential answers. WordNet (Fellbaum, 1998) is perhaps the most popular resource and has been employed in a variety of QA-related tasks ranging from query expansion, to axiom-based reasoning (Moldovan et al., 2003), passage scoring (Paranjpe et al., 2003), and answer filtering (Leidner et al., 2004). Besides WordNet, recent QA systems increasingly rely on syntactic information as a means of abstracting over word order differences and structural alternations (e.g., passive vs. active voice). Most syntax-based QA systems (Wu et al., 2005) incorporate some means of comparison between the tree representing the question with the subtree surrounding the answer candidate. The assumption here is that appropriate answers are more likely to have syntactic relations in common with their corresponding question. Syntactic structure matching has been applied to passage retrieval (Cui et al., 2005) and answer extraction (Shen and Klakow, 2006).

Narayanan and Harabagiu (2004) were the first to stress the importance of semantic roles in answering complex questions. Their system identifies predicate argument structures by merging semantic role information from PropBank and FrameNet. Expected answers are extracted by performing probabilistic inference over the predicate argument structures in conjunction with a domain specific topic model. Sun et al. (2005) incorporate semantic analysis in their TREC05 QA system. They use ASSERT (Pradhan et al., 2004), a publicly available shallow semantic parser trained on PropBank, to generate predicate-argument structures which subsequently form the basis of comparison between question and answer sentences. They find that semantic analysis does not boost performance due to the low recall of the semantic parser. Kaisser (2006) proposes a

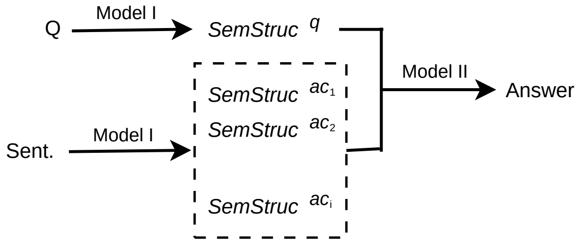


Figure 1: Architecture of answer extraction

question paraphrasing method based on FrameNet. Questions are assigned semantic roles by matching their dependency relations with those attested in the FrameNet annotations. The assignments are used to create question reformulations which are submitted to Google for answer extraction. The semantic role assignment module is not probabilistic, it relies on strict matching, and runs into severe coverage problems.

In line with previous work, our method exploits syntactic information in the form of dependency relation paths together with FrameNet-like semantic roles to smooth lexical and syntactic divergences between question and answer sentences. Our approach is less domain dependent and resource intensive than Narayanan and Harabagiu (2004), it solely employs a dependency parser and the FrameNet database. In contrast to Kaisser (2006), we model the semantic role assignment and answer extraction tasks numerically, thereby alleviating the coverage problems encountered previously.

3 Problem Formulation

We briefly summarize the architecture of the QA system we are working with before formalizing the mechanics of our FrameNet-based answer extraction module. In common with previous work, our overall approach consists of three stages: (a) determining the expected answer type of the question, (b) retrieving passages likely to contain answers to the question, and (c) performing a match between the question words and retrieved passages in order to extract the answer. In this paper we focus on the last stage: question and answer sentences are normalized to a FrameNet-style representation and answers are retrieved by selecting the candidate whose semantic structure is most similar to the question.

The architecture of our answer extraction module is shown in Figure 1. Semantic structures for questions and sentences are automatically derived using the model described in Section 4 (Model I). A semantic structure $SemStruc = \langle p, Set(SRA) \rangle$ consists of a predicate p and a set of semantic role assignments $Set(SRA)$. p is a word or phrase evoking a frame F of FrameNet. A semantic role assignment SRA is a ternary structure $\langle w, SR, s \rangle$, consisting of frame element w, its semantic role SR, and score s indicating to what degree SR qualifies as a label for w.

For a question q, we generate a semantic structure $SemStruc^q$. Question words, such as *what*, *who*, *when*, etc., are considered expected answer phrases (*EAP*s). We require that *EAP*s are frame elements of $SemStruc^q$. Likely answer candidates are extracted from answer sentences following some preprocessing steps detailed in Section 6. For each candidate ac, we derive its semantic structure $SemStruc^{ac}$ and assume that ac is a frame element of $SemStruc^{ac}$. Question and answer semantic structures are compared using a model based on graph matching detailed in Section 5 (Model II). We calculate the similarity of all derived pairs $\langle SemStruc^q, SemStruc^{ac} \rangle$ and select the candidate with the highest value as an answer for the question.

4 Semantic Structure Generation

Our method crucially exploits the annotated sentences in the FrameNet database together with the output of a dependency parser. Our guiding assumption is that sentences that share dependency relations will also share semantic roles as long as they evoke the same or related frames. This is motivated by much research in lexical semantics (e.g., Levin (1993)) hypothesizing that the behavior of words, particularly with respect to the expression and interpretation of their arguments, is to a large extent determined by their meaning. We first describe how predicates are identified and then introduce our model for semantic role labeling.

Predicate Identification Predicate candidates are identified using a simple look-up procedure which compares POS-tagged tokens against FrameNet entries. For efficiency reasons, we make the simplifying assumption that questions have only one predicate which we select heuristically: (1) verbs are preferred to other parts of speech, (2) if there is more than one verb in the question, preference is given to the verb with the highest level of embedding in the dependency tree, (3) if no verbs are present, a noun is chosen. For example, in *Q: Who beat Floyd Patterson to take the title away?*, *beat*, *take away*, and *title* are identified as predicate candidates and *beat* is selected the main predicate of the question. For answer sentences, we require that the predicate is either identical or semantically related to the question predicate (see Section 5).

In the example given above, the predicate *beat* evoques a single frame (i.e., *Cause_harm*). However, predicates often have multiple meanings thus evoquing more than one frame. Knowing which is the appropriate frame for a given predicate impacts the semantic role assignment task; selecting the wrong frame will unavoidably result in erroneous semantic roles. Rather than disambiguiting polysemous predicates prior to semantic role assignment, we perform the assignment for each frame evoqued by the predicate.

Semantic Role Assignment Before describing our approach to semantic role labeling we define dependency relation paths. A relation path R is a relation sequence $\langle r_1, r_2, ..., r_L \rangle$, in which r_l ($l = 1, 2, ..., L$) is one of predefined dependency relations with suffix of traverse direction. An example of a relation path is $R = \langle subj_U, obj_D \rangle$, where the subscripts U and D indicate upward and downward movement in trees, respectively. Given an unannotated sentence whose roles we wish to label, we assume that words or phrases w with a dependency path connecting them to p are frame elements. Each frame element is represented by an *unlabeled* dependency path R_w which we extract by traversing the dependency tree from w to p. Analogously, we extract from the FrameNet annotations all dependency paths R_{SR} that are *labeled* with semantic role information and correspond to p. We next measure the compatibility of labeled and unlabeled paths as follows:

(2) $$s(w, SR) = \max_{R_{SR} \in M} [sim(R_w, R_{SR}) \cdot P(R_{SR})]$$

where M is the set of dependency relation paths for SR in FrameNet, $sim(R_w, R_{SR})$ the similarity between paths R_w and R_{SR} weighted by the relative frequency of R_{SR} in FrameNet ($P(R_{SR})$). We consider both core and non-core semantic roles instantiated by frames with at least one annotation in FrameNet. Core roles tend to have more annotations in Framenet and consequently are considered more probable.

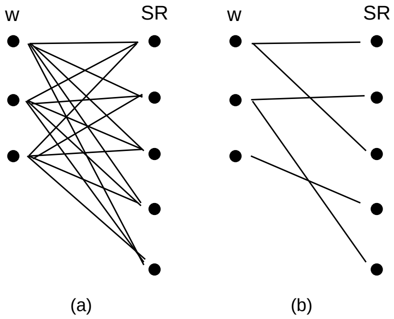


Figure 2: Sample original bipartite graph (a) and its subgraph with edge covers (b). In each graph, the left partition represents frame elements and the right partition semantic roles.

We measure $sim(R_w, R_{SR})$, by adapting a string kernel to our task. Our hypothesis is that the more common substrings two dependency paths have, the more similar they are. The string kernel we used is similar to Leslie (2002) and defined as the sum of weighted common dependency relation subsequences between R_w and R_{SR}. For efficiency, we consider only unigram and bigram subsequences. Subsequences are weighted by a metric akin to $tf \cdot idf$ which measures the degree of association between a candidate SR and the dependency relation r present in the subsequence:

(3) $$weight_{SR}(r) = f_r \cdot \log\left(1 + \frac{N}{n_r}\right)$$

where f_r is the frequency of r occurring in SR; N is the total number of SRs evoked by a given frame; and n_r is the number of SRs containing r.

For each frame element we thus generate a set of semantic role assignments $Set(SRA)$. This initial assignment can be usefully represented as a complete bipartite graph in which each frame element (word or phrase) is connected to the semantic roles licensed by the predicate and vice versa. (see Figure 2a). Edges are weighted and represent how compatible the frame elements and semantic roles are (see equation (2)). Now, for each frame element w

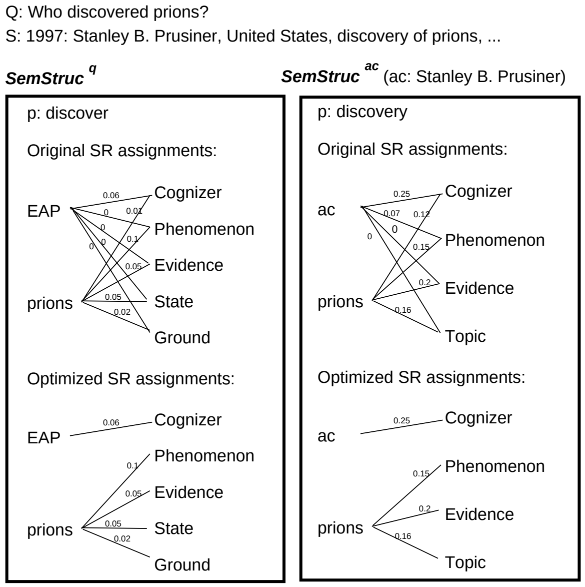


Figure 3: Semantic structures induced by our model for a question and answer sentence

we could simply select the semantic role with the highest score. However, this decision procedure is *local*, i.e., it yields a semantic role assignment for each frame element independently of all other elements. We therefore may end up with the same role being assigned to two frame elements or with frame elements having no role at all. We remedy this shortcoming by treating the semantic role assignment as a *global* optimization problem.

Specifically, we model the interaction between *all* pairwise labeling decisions as a *minimum weight bipartite edge cover problem* (Eiter and Mannila, 1997; Cormen et al., 1990). An edge cover is a subgraph of a bipartite graph so that each node is linked to at least one node of the other partition. This yields a semantic role assignment for all frame elements (see Figure 2b where frame elements and roles are adjacent to an edge). Edge covers have been successfully applied in several natural language processing tasks, including machine translation (Taskar et al., 2005) and annotation projection (Padó and Lapata, 2006).

Formally, optimal edge cover assignments are solutions of following optimization problem:

(4)
$$\max_{E \text{ is edge cover}} \prod_{(nd^w, nd^{SR}) \in E} s(nd^w, nd^{SR})$$

where, $s(nd^w, nd^{SR})$ is the compatibility score between the frame element node nd^w and semantic role node nd^{SR}. Edge covers can be computed efficiently in cubic time using algorithms for the equivalent linear assignment problem. Our experiments used Jonker and Volgenant's (1987) solver.[2]

Figure 3 shows the semantic role assignments generated by our model for the question *Q: Who discovered prions?* and the candidate answer sentence *S: 1997: Stanley B. Prusiner, United States, discovery of prions...* Here we identify two predicates, namely *discover* and *discovery*. The expected answer phrase (EAP) *who* and the answer candidate *Stanley B. Prusiner* are assigned the COGNIZER role. Note that frame elements can bear multiple semantic roles. By inducing a soft labeling we hope to render the matching of questions and answers more robust, thereby addressing to some extent the coverage problems associated with FrameNet.

5 Semantic Structure Matching

We measure the similarity between a question and its candidate answer by matching their predicates and semantic role assignments. Since SRs are frame-specific, we prioritize frame matching to SR matching. Two predicates match if they evoke the same frame or one of its hypernyms (or hyponyms). The latter are expressed by the *Inherits From* and *Is Inherited By* relations in the frame definitions. If the predicates match, we examine whether the assigned semantic roles match. Since we represent SR assignments as graphs with edge covers, we can also formalize SR matching as a graph matching problem.

The similarity between two graphs is measured as the sum of similarities between their subgraphs. We first decompose a graph into subgraphs consisting of one frame element node w and a set of SR nodes connected to it. The similarity between two subgraphs $SubG_1$, and $SubG_2$ is then formalized as:

(5)
$$Sim(SubG_1, SubG_2) = \sum_{\substack{nd_1^{SR} \in SubG_1 \\ nd_2^{SR} \in SubG_2 \\ nd_1^{SR} = nd_2^{SR}}} \frac{1}{|s(nd^w, nd_1^{SR}) - s(nd^w, nd_2^{SR})| + 1}$$

where, nd_1^{SR} and nd_2^{SR} are semantic role nodes connected to a frame element node nd^w in $SubG_1$ and

[2] The software is available from `http://www.magiclogic.com/assignment.html` .

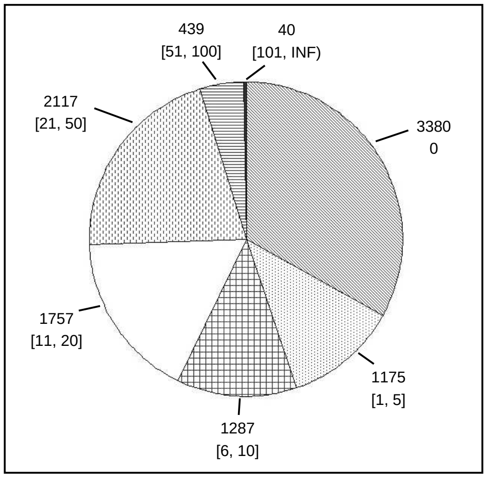


Figure 4: Distribution of Numbers of Predicates and annotated sentences; each sub-pie, lists the number of predicates (above) with their corresponding range of annotated sentences (below)

$SubG_2$, respectively. $s(nd^w, nd_1^{sr})$ and $s(nd^w, nd_2^{SR})$ are edge weights between two nodes in corresponding subgraphs (see (2)). Our intuition here is that the more semantic roles two subgraphs share for a given frame element, the more similar they are and the closer their corresponding edge weights should be. Edge weights are normalized by dividing by the sum of all edges in a subgraph.

6 Experimental Setup

Data All our experiments were performed on the TREC02–05 factoid questions. We excluded NIL questions since TREC doesn't supply an answer for them. We used the FrameNet V1.3 lexical database. It contains 10,195 predicates grouped into 795 semantic frames and 141,238 annotated sentences. Figure 4 shows the number of annotated sentences available for different predicates. As can be seen, there are 3,380 predicates with no annotated sentences and 1,175 predicates with less than 5 annotated sentences. All FrameNet sentences, questions, and answer sentences were parsed using MiniPar (Lin, 1994), a robust dependency parser.

As mentioned in Section 4 we extract dependency relation paths by traversing the dependency tree from the frame element node to the predicate node. We used all dependency relations provided by MiniPar (42 in total). In order to increase coverage, we combine all relation paths for predicates that evoke the same frame and are labeled with the same POS tag. For example, *found* and *establish* are both instances of the frame *Intentionally_create* but the database does not have any annotated sentences for *found.v*. In default of not assigning any role labels for *found.v*, our model employs the relation paths for the semantically related *establish.v*.

Preprocessing Here we summarize the steps of our QA system preceding the assignment of semantic structure and answer extraction. For each question, we recognize its expected answer type (e.g., in *Q: Which record company is Fred Durst with?* we would expect the answer to be an *ORGANIZATION*). Answer types are determined using classification rules similar to Li and Roth (2002). We also reformulate questions into declarative sentences following the strategy proposed in Brill et al. (2002).

The reformulated sentences are submitted as queries to an IR engine for retrieving sentences with relevant answers. Specifically, we use the Lemur Toolkit[3], a state-of-the-art language model-driven search engine. We work only with the 50 top-ranked sentences as this setting performed best in previous experiments of our QA system. We also add to Lemur's output gold standard sentences, which contain and support an answer for each question. Specifically, documents relevant for each question are retrieved from the AQUAINT Corpus[4] according to TREC supplied judgments. Next, sentences which match both the TREC provided answer pattern and at least one question key word are extracted and their suitability is manually judged by humans. The set of relevant sentences thus includes at least one sentence with an appropriate answer as well as sentences that do not contain any answer specific information. This setup is somewhat idealized, however it allows us to evaluate in more detail our answer extraction module (since when an answer is not found, we know it is the fault of our system).

Relevant sentences are annotated with their named entities using Lingpipe[5], a MUC-based named entity recognizer. When we successfully classify a question with an expected answer type

[3] See http://www.lemurproject.org/ for details.

[4] This corpus consists of English newswire texts and is used as the main document collection in official TREC evaluations.

[5] The software is available from www.alias-i.com/lingpipe/

(e.g., *ORGANIZATION* in the example above), we assume that all NPs attested in the set of relevant sentences with the same answer type are candidate answers; in cases where no answer type is found (e.g., as in *Q: What are prions made of?*), all NPs in the relevant answers set are considered candidate answers.

Baseline We compared our answer extraction method to a QA system that exploits solely syntactic information without making use of FrameNet or any other type of role semantic annotations. For each question, the baseline identifies key phrases deemed important for answer identification. These are verbs, noun phrases, and expected answer phrases (EAPs, see Section 3). All dependency relation paths connecting a key phrase and an EAP are compared to those connecting the same key phrases and an answer candidate. The similarity of question and answer paths is computed using a simplified version of the similarity measure[6] proposed in Shen and Klakow (2006).

Our second baseline employs Shalmaneser (Erk and Padó, 2006), a publicly available shallow semantic parser[7], for the role labeling task instead of the graph-based model presented in Section 4. The software is trained on the FrameNet annotated sentences using a standard feature set (see Carreras and Màrquez (2005) for details). We use Shalmaneser to parse questions and answer sentences. The parser makes hard decisions about the presence or absence of a semantic role. Unfortunately, this prevents us from using our method for semantic structure matching (see Section 5) which assumes a soft labeling. We therefore came up with a simple matching strategy suitable for the parser's output. For question and answer sentences matching in their frame assignment, phrases bearing the same semantic role as the EAP are considered answer candidates. The latter are ranked according to word overlap (i.e., identical phrases are ranked higher than phrases with no overlap at all).

7 Results

Our evaluation was motivated by the following questions: (1) How does the incompleteness of FrameNet impact QA performance on the TREC data sets? In particular, we wanted to examine whether there are questions for which in principle no answer can be found due to missing frame entries or missing annotated sentences. (2) Are all questions and their corresponding answers amenable to a FrameNet-style analysis? In other words, we wanted to assess whether questions and answers often evoke the same or related frames (with similar roles). This is a prerequisite for semantic structure matching and ultimately answer extraction. (3) Do the graph-based models introduced in this paper bring any performance gains over state-of-the-art shallow semantic parsers or more conventional syntax-based QA systems? Recall that our graph-based models were designed especially for the QA answer extraction task.

Our results are summarized in Tables 1–3. Table 1 records the number of questions to be answered for the TREC02–05 datasets (Total). We also give information regarding the number of questions which are in principle *unanswerable* with a FrameNet-style semantic role analysis.

Column NoFrame shows the number of questions which don't have an appropriate frame or predicate in the database. For example, there is currently no predicate entry for *sponsor* or *sink* (e.g., *Q: Who is the sponsor of the International Criminal Court?* and *Q: What date did the Lusitania sink?*). Column NoAnnot refers to questions for which no semantic role labeling is possible because annotated sentences for the relevant predicates are missing. For instance, there are no annotations for *win* (e.g., *Q: What division did Floyd Patterson win?*) or for *hit* (e.g., *Q: What was the Beatles' first number one hit?*). This problem is not specific to our method which admittedly relies on FrameNet annotations for performing the semantic role assignment (see Section 4). Shallow semantic parsers trained on FrameNet would also have trouble assigning roles to predicates for which no data is available.

Finally, column NoMatch reports the number of questions which cannot be answered due to frame

[6] Shen and Klakow (2006) use a dynamic time warping algorithm to calculate the degree to which dependency relation paths are correlated. Correlations for individual relations are estimated from training data whereas we assume a binary value (1 for identical relations and 0 otherwise). The modification was necessary to render the baseline system comparable to our answer extraction model which is unsupervised.

[7] The software is available from http://www.coli.uni-saarland.de/projects/salsa/shal/ .

Data	Total	NoFrame		NoAnnot		NoMatch		Rest	
TREC02	444	87	(19.6)	29	(6.5)	176	(39.6)	152	(34.2)
TREC03	380	55	(14.5)	30	(7.9)	183	(48.2)	112	(29.5)
TREC04	203	47	(23.1)	14	(6.9)	67	(33.0)	75	(36.9)
TREC05	352	70	(19.9)	23	(6.5)	145	(41.2)	114	(32.4)

Table 1: Number of questions which cannot be answered using a FrameNet style semantic analysis; numbers in parentheses are percentages of Total (NoFrame: frames or predicates are missing; NoAnnot: annotated sentences are missing, NoMatch: questions and candidate answers evoke different frames.

mismatches. Consider *Q: What does AARP stand for?* whose answer is found in *S: The American Association of Retired Persons (AARP) qualify for discounts…*. The answer and the question evoke different frames; in fact here a semantic role analysis is not relevant for locating the right answer. As can be seen NoMatch cases are by far the most frequent. The number of questions remaining after excluding NoFrame, NoAnnot, and NoMatch are shown under the Rest heading in Table 1.

These results indicate that FrameNet-based semantic role analysis applies to approximately 35% of the TREC data. This means that an extraction module relying solely on FrameNet will have poor performance, since it will be unable to find answers for more than half of the questions beeing asked. We nevertheless examine whether our model brings any performance improvements on this limited dataset which is admittedly favorable towards a FrameNet style analysis. Table 2 shows the results of our answer extraction module (SemMatch) together with two baseline systems. The first baseline uses only dependency relation path information (SynMatch), whereas the second baseline (SemParse) uses Shalmaneser, a state-of-the-art shallow semantic parser for the role labeling task. We consider an answer correct if it is returned with rank 1. As can be seen, SemMatch is significantly better than both SynMatch and SemParse, whereas the latter is significantly worse than SynMatch.

Model	TREC02	TREC03	TREC04	TREC05
SemParse	13.16	8.92	17.33	13.16
SynMatch	35.53*	33.04*	40.00*	36.84*
SemMatch	53.29*†	49.11*†	54.67*†	59.65*†

Table 2: System Performance on subset of TREC datasets (see Rest column in Table 1); *: significantly better than SemParse; †: significantly better than SynMatch ($p < 0.01$, using a χ^2 test).

Model	TREC02	TREC03	TREC04	TREC05
SynMatch	32.88*	30.70*	35.95*	34.38*
+SemParse	25.23	23.68	28.57	26.70
+SemMatch	38.96*†	35.53*†	42.36*†	41.76*†

Table 3: System Performance on TREC datasets (see Total column in Table 1); *: significantly better than +SemParse; †: significantly better than SynMatch ($p < 0.01$, using a χ^2 test).

Although promising, the results in Table 2 are not very informative, since they show performance gains on partial data. Instead of using our answer extraction model on its own, we next combined it with the syntax-based system mentioned above (SynMatch, see also Section 6 for details). If FrameNet is indeed helpful for QA, we would expect an ensemble system to yield better performance over a purely syntactic answer extraction module. The two systems were combined as follows. Given a question, we first pass it to our FrameNet model; if an answer is found, our job is done; if no answer is returned, the question is passed on to SynMatch. Our results are given in Table 3. +SemMatch and +SemParse are ensemble systems using SynMatch together with the QA specific role labeling method proposed in this paper and Shalmaneser, respectively. We also compare these systems against SynMatch on its own.

We can now attempt to answer our third question concerning our model's performance on the TREC data. Our experiments show that a FrameNet-enhanced answer extraction module significantly outperforms a similar module that uses only syntactic information (compare SynMatch and +SemMatch in Table 3). Another interesting finding is that the shallow semantic parser performs considerably worse in comparison to our graph-based models and the syntax-based system. Inspection of the parser's output highlights two explanations for this. First, the shallow semantic parser has difficulty assigning accurate semantic roles to questions (even when they are reformulated as declarative sentences). And secondly, it tends to favor precision over recall, thus reducing the number of questions for which answers can be found. A similar finding is reported in Sun et al. (2005) for a PropBank trained parser.

8 Conclusion

In this paper we assess the contribution of semantic role labeling to open-domain factoid question answering. We present a graph-based answer extraction model which effectively incorporates FrameNet style role semantic information and show that it achieves promising results. Our experiments show that the proposed model can be effectively combined with a syntax-based system to obtain performance superior to the latter when used on its own. Furthermore, we demonstrate performance gains over a shallow semantic parser trained on the FrameNet annotated corpus. We argue that performance gains are due to the adopted graph-theoretic framework which is robust to coverage and recall problems.

We also provide a detailed analysis of the appropriateness of FrameNet for QA. We show that performance can be compromised due to incomplete coverage (i.e., missing frame or predicate entries as well as annotated sentences) but also because of mismatching question-answer representations. The question and the answer may evoke different frames or the answer simply falls outside the scope of a given frame (i.e., in a non predicate-argument structure). Our study shows that mismatches are relatively frequent and motivates the use of semantically informed methods in conjunction with syntax-based methods.

Important future directions lie in evaluating the contribution of alternative semantic role frameworks (e.g., PropBank) to the answer extraction task and developing models that learn semantic roles directly from unannotated text without the support of FrameNet annotations (Grenager and Manning, 2006). Beyond question answering, we also plan to investigate the potential of our model for shallow semantic parsing since our experience so far has shown that it achieves good recall.

Acknowledgements We are grateful to Sebastian Padó for running Shalmaneser on our data. Thanks to Frank Keller and Amit Dubey for insightful comments and suggestions. The authors acknowledge the support of DFG (Shen; PhD studentship within the International Postgraduate College "Language Technology and Cognitive Systems") and EPSRC (Lapata; grant EP/C538447/1).

References

E. Brill, S. Dumais, M. Banko. 2002. An analysis of the askMSR question-answering system. In *Proceedings of the EMNLP*, 257–264, Philadelphia, PA.

X. Carreras, L. Màrquez, eds. 2005. *Proceedings of the CoNLL shared task: Semantic role labelling*, 2005.

T. Cormen, C. Leiserson, R. Rivest. 1990. *Introduction to Algorithms*. MIT Press.

H. Cui, R. X. Sun, K. Y. Li, M. Y. Kan, T. S. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the ACM SIGIR*, 400–407. ACM Press.

T. Eiter, H. Mannila. 1997. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133.

K. Erk, S. Padó. 2006. Shalmaneser - a flexible toolbox for semantic role assignment. In *Proceedings of the LREC*, 527–532, Genoa, Italy.

C. Fellbaum, ed. 1998. *WordNet. An Electronic Lexical Database*. MIT Press, Cambridge/Mass.

C. J. Fillmore, C. R. Johnson, M. R. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250.

D. Gildea, D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

T. Grenager, C. D. Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the EMNLP*, 1–8, Sydney, Australia.

R. Jonker, A. Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340.

M. Kaisser. 2006. Web question answering by exploiting wide-coverage lexical resources. In *Proceedings of the 11th ESSLLI Student Session*, 203–213.

J. Leidner, J. Bos, T. Dalmas, J. Curran, S. Clark, C. Bannard, B. Webber, M. Steedman. 2004. The qed open-domain answer retrieval system for TREC 2003. In *Proceedings of the TREC*, 595–599.

C. Leslie, E. Eskin, W. S. Noble. 2002. The spectrum kernel: a string kernel for SVM protein classification. In *Proceedings of the Pacific Biocomputing Symposium*, 564–575.

B. Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago.

X. Li, D. Roth. 2002. Learning question classifiers. In *Proceedings of the 19th COLING*, 556–562, Taipei, Taiwan.

D. K. Lin. 1994. PRINCIPAR–an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th COLING*, 482–488.

D. Moldovan, C. Clark, S. Harabagiu, S. Maiorano. 2003. COGEX: A logic prover for question answering. In *Proceedings of the HLT/NAACL*, 87–93, Edmonton, Canada.

S. Narayanan, S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 19th COLING*, 184–191.

S. Padó, M. Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the COLING/ACL*, 1161–1168.

M. Palmer, D. Gildea, P. Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

D. Paranjpe, G. Ramakrishnan, S. Srinivasa. 2003. Passage scoring for question answering via bayesian inference on lexical relations. In *Proceedings of the TREC*, 305–210.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the HLT/NAACL*, 141–144, Boston, MA.

D. Shen, D. Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the COLING/ACL*, 889–896.

R. X. Sun, J. J. Jiang, Y. F. Tan, H. Cui, T. S. Chua, M. Y. Kan. 2005. Using syntactic and semantic relation analysis in question answering. In *Proceedings of the TREC*.

B. Taskar, S. Lacoste-Julien, D. Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of the HLT/EMNLP*, 73–80, Vancouver, BC.

M. Wu, M. Y. Duan, S. Shaikh, S. Small, T. Strzalkowski. 2005. University at albany's ilqua in trec 2005. In *Proceedings of the TREC*, 77–83.

What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA

Mengqiu Wang and **Noah A. Smith** and **Teruko Mitamura**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{mengqiu,nasmith,teruko}@cs.cmu.edu

Abstract

This paper presents a syntax-driven approach to question answering, specifically the answer-sentence selection problem for short-answer questions. Rather than using syntactic features to augment existing statistical classifiers (as in previous work), we build on the idea that questions and their (correct) answers relate to each other via loose but predictable syntactic transformations. We propose a probabilistic quasi-synchronous grammar, inspired by one proposed for machine translation (D. Smith and Eisner, 2006), and parameterized by mixtures of a robust nonlexical syntax/alignment model with a(n optional) lexical-semantics-driven log-linear model. Our model learns soft alignments as a hidden variable in discriminative training. Experimental results using the TREC dataset are shown to significantly outperform strong state-of-the-art baselines.

1 Introduction and Motivation

Open-domain question answering (QA) is a widely-studied and fast-growing research problem. State-of-the-art QA systems are extremely complex. They usually take the form of a pipeline architecture, chaining together modules that perform tasks such as answer type analysis (identifying whether the correct answer will be a person, location, date, etc.), document retrieval, answer candidate extraction, and answer reranking. This architecture is so predominant that each task listed above has evolved into its own sub-field and is often studied and evaluated independently (Shima et al., 2006).

At a high level, the QA task boils down to only two essential steps (Echihabi and Marcu, 2003). The first step, **retrieval**, narrows down the search space from a corpus of millions of documents to a focused set of maybe a few hundred using an IR engine, where efficiency and recall are the main focus. The second step, **selection**, assesses each candidate answer string proposed by the first step, and finds the one that is most likely to be an answer to the given question. The granularity of the target answer string varies depending on the type of the question. For example, answers to factoid questions (e.g., Who, When, Where) are usually single words or short phrases, while definitional questions and other more complex question types (e.g., How, Why) look for sentences or short passages. In this work, we fix the granularity of an answer to a single sentence.

Earlier work on answer selection relies only on the surface-level text information. Two approaches are most common: surface pattern matching, and similarity measures on the question and answer, represented as bags of words. In the former, patterns for a certain answer type are either crafted manually (Soubbotin and Soubbotin, 2001) or acquired from training examples automatically (Ittycheriah et al., 2001; Ravichandran et al., 2003; Licuanan and Weischedel, 2003). In the latter, measures like cosine-similarity are applied to (usually) bag-of-words representations of the question and answer. Although many of these systems have achieved very good results in TREC-style evaluations, shallow methods using the bag-of-word representation clearly have their limitations. Examples of

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 22–32, Prague, June 2007.

cases where the bag-of-words approach fails abound in QA literature; here we borrow an example used by Echihabi and Marcu (2003). The question is "*Who is the leader of France?*", and the sentence "Henri Hadjenberg, *who is the leader of France* 's Jewish community, endorsed ..." (note tokenization), which is not the correct answer, matches all keywords in the question in exactly the same order. (The correct answer is found in "Bush later met with French President Jacques Chirac.")

This example illustrates two types of variation that need to be recognized in order to connect this question-answer pair. The first variation is the change of the word "leader" to its semantically related term "president". The second variation is the syntactic shift from "leader of France" to "French president." It is also important to recognize that "France" in the first sentence is modifying "community", and therefore "Henri Hadjenberg" is the "leader of ... community" rather than the "leader of France." These syntactic and semantic variations occur in almost every question-answer pair, and typically they cannot be easily captured using shallow representations. It is also worth noting that such syntactic and semantic variations are not unique to QA; they can be found in many other closely related NLP tasks, motivating extensive community efforts in syntactic and semantic processing.

Indeed, in this work, we imagine a generative story for QA in which the question is generated from the answer sentence through a series of syntactic and semantic transformations. The same story has been told for **machine translation** (Yamada and Knight, 2001, *inter alia*), in which a target language sentence (the desired output) has undergone semantic transformation (word to word translation) and syntactic transformation (syntax divergence across languages) to generate the source language sentence (noisy-channel model). Similar stories can also be found in **paraphrasing** (Quirk et al., 2004; Wu, 2005) and **textual entailment** (Harabagiu and Hickl, 2006; Wu, 2005).

Our story makes use of a weighted formalism known as **quasi-synchronous grammar** (hereafter, QG), originally developed by D. Smith and Eisner (2006) for machine translation. Unlike most synchronous formalisms, QG does not posit a strict isomorphism between the two trees, and it provides an elegant description for the set of local configurations. In Section 2 we situate our contribution in the context of earlier work, and we give a brief discussion of quasi-synchronous grammars in Section 3. Our version of QG, called the **Jeopardy model**, and our parameter estimation method are described in Section 4. Experimental results comparing our approach to two state-of-the-art baselines are presented in Section 5. We discuss portability to cross-lingual QA and other applied semantic processing tasks in Section 6.

2 Related Work

To model the syntactic transformation process, researchers in these fields—especially in machine translation—have developed powerful grammatical formalisms and statistical models for representing and learning these tree-to-tree relations (Wu and Wong, 1998; Eisner, 2003; Gildea, 2003; Melamed, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Galley et al., 2006; Smith and Eisner, 2006, *inter alia*). We can also observe a trend in recent work in textual entailment that more emphasis is put on explicit learning of the syntactic graph mapping between the entailed and entailed-by sentences (MacCartney et al., 2006).

However, relatively fewer attempts have been made in the QA community. As pointed out by Katz and Lin (2003), most early experiments in QA that tried to bring in syntactic or semantic features showed little or no improvement, and it was often the case that performance actually degraded (Litkowski, 1999; Attardi et al., 2001). More recent attempts have tried to augment the bag-of-words representation—which, after all, is simply a real-valued feature vector—with syntactic features. The usual similarity measures can then be used on the new feature representation. For example, Punyakanok et al. (2004) used approximate tree matching and tree-edit-distance to compute a similarity score between the question and answer parse trees. Similarly, Shen et al. (2005) experimented with dependency tree kernels to compute similarity between parse trees. Cui et al. (2005) measured sentence similarity based on similarity measures between dependency paths among aligned words. They used heuristic functions similar to mutual information to assign scores to matched pairs of dependency links. Shen and Klakow (2006) extend the idea further through the use of log-linear models to learn a scoring function for relation pairs.

Echihabi and Marcu (2003) presented a noisy-channel approach in which they adapted the IBM model 4 from statistical machine translation (Brown et al., 1990; Brown et al., 1993) and applied it to QA. Similarly, Murdock and Croft (2005) adopted a simple translation model from IBM model 1 (Brown et al., 1990; Brown et al., 1993) and applied it to QA. Porting the translation model to QA is not straightforward; it involves parse-tree pruning heuristics (the first two deterministic steps in Echihabi and Marcu, 2003) and also replacing the lexical translation table with a monolingual "dictionary" which simply encodes the identity relation. This brings us to the question that drives this work: is there a statistical translation-like model that is natural and accurate for question answering? We propose Smith and Eisner's (2006) quasi-synchronous grammar (Section 3) as a general solution and the Jeopardy model (Section 4) as a specific instance.

3 Quasi-Synchronous Grammar

For a formal description of QG, we recommend Smith and Eisner (2006). We briefly review the central idea here. QG arose out of the empirical observation that translated sentences often have *some* isomorphic syntactic structure, but not usually in entirety, and the strictness of the isomorphism may vary across words or syntactic rules. The idea is that, rather than a synchronous structure over the source and target sentences, a tree over the target sentence is modeled by a source-sentence-specific grammar that is inspired by the source sentence's tree.[1] This is implemented by a "sense"—really just a subset of nodes in the *source* tree—attached to each grammar node in the *target* tree. The senses define an alignment between the trees. Because it only loosely links the two sentences' syntactic structure, QG is particularly well-suited for QA insofar as QA is like "free" translation.

A concrete example that is easy to understand is a binary quasi-synchronous context-free grammar (denoted QCFG). Let V_S be the set of constituent tokens in the source tree. QCFG rules would take the augmented form

$$\begin{array}{rcl} \langle X, \mathbb{S}_1 \rangle & \rightarrow & \langle Y, \mathbb{S}_2 \rangle \langle Z, \mathbb{S}_3 \rangle \\ \langle X, \mathbb{S}_1 \rangle & \rightarrow & w \end{array}$$

where X, Y, and Z are ordinary CFG nonterminals, each $\mathbb{S}_i \in 2^{V_S}$ (subsets of nodes in the source tree to which the nonterminals align), and w is a target-language word. QG can be made more or less "liberal" by constraining the cardinality of the $\mathbb{S}_i$ (we force all $|\mathbb{S}_i| = 1$), and by constraining the relationships *among* the $\mathbb{S}_i$ mentioned in a single rule. These are called permissible "configurations." An example of a strict configuration is that a target parent-child pair must align (respectively) to a *source* parent-child pair. Configurations are shown in Table 1.

Here, following Smith and Eisner (2006), we use a weighted, quasi-synchronous *dependency* grammar. Apart from the obvious difference in application task, there are a few important differences with their model. First, we are not interested in the alignments *per se*; we will sum them out as a hidden variable when scoring a question-answer pair. Second, our probability model includes an optional mixture component that permits arbitrary features—we experiment with a small set of WordNet lexical-semantics features (see Section 4.4). Third, we apply a more discriminative training method (conditional maximum likelihood estimation, Section 4.5).

4 The Jeopardy Model

Our model, informally speaking, aims to follow the process a player of the television game show *Jeopardy!* might follow. The player knows the answer (or at least *thinks* he knows the answer) and must quickly turn it into a question.[2] The question-answer pairs used on *Jeopardy!* are not precisely what we have in mind for the real task (the questions are not specific enough), but the syntactic transformation inspires our model. In this section we formally define

[1] Smith and Eisner also show how QG formalisms generalize synchronous grammar formalisms.

[2] A round of *Jeopardy!* involves a somewhat involved and specific "answer" presented to the competitors, and the first competitor to hit a buzzer proposes the "question" that leads to the answer. For example, an answer might be, *This Eastern European capital is famous for defenestrations.* In *Jeopardy!* the players must respond with a queston: *What is Prague?*

this probability model and present the necessary algorithms for parameter estimation.

4.1 Probabilistic Model

The Jeopardy model is a QG designed for QA. Let $\mathbf{q} = \langle q_1, ..., q_n \rangle$ be a question sentence (each q_i is a word), and let $\mathbf{a} = \langle a_1, ..., a_m \rangle$ be a candidate answer sentence. (We will use $\mathbf{w}$ to denote an abstract sequence that could be a question or an answer.) In practice, these sequences may include other information, such as POS, but for clarity we assume just words in the exposition. Let $\mathcal{A}$ be the set of candidate answers under consideration. Our aim is to choose:

$$\hat{\mathbf{a}} = \operatorname*{argmax}_{\mathbf{a} \in \mathcal{A}} p(\mathbf{a} \mid \mathbf{q}) \tag{1}$$

At a high level, we make three adjustments. The first is to apply Bayes' rule, $p(\mathbf{a} \mid \mathbf{q}) \propto p(\mathbf{q} \mid \mathbf{a}) \cdot p(\mathbf{a})$. Because $\mathcal{A}$ is known and is assumed to be generated by an external extraction system, we could use that extraction system to assign scores (and hence, probabilities $p(\mathbf{a})$) to the candidate answers. Other scores could also be used, such as reputability of the document the answer came from, grammaticality, etc. Here, aiming for simplicity, we do not aim to use such information. Hence we treat $p(\mathbf{a})$ as uniform over $\mathcal{A}$.[3]

The second adjustment adds a labeled, directed **dependency tree** to the question and the answer. The tree is produced by a state-of-the-art dependency parser (McDonald et al., 2005) trained on the *Wall Street Journal* Penn Treebank (Marcus et al., 1993). A dependency tree on a sequence $\mathbf{w} = \langle w_1, ..., w_k \rangle$ is a mapping of indices of words to indices of their syntactic parents and a label for the syntactic relation, $\tau : \{1, ..., k\} \rightarrow \{0, ..., k\} \times \mathcal{L}$. Each word w_i has a single parent, denoted $w_{\tau(i).par}$. Cycles are not permitted. w_0 is taken to be the invisible "wall" symbol at the left edge of the sentence; it has a single child ($|\{i : \tau(i) = 0\}| = 1$). The label for w_i is denoted $\tau(i).lab$.

The third adjustment involves a hidden variable X, the **alignment** between question and answer words. In our model, each question-word maps to exactly one answer-word. Let $x : \{1, ..., n\} \rightarrow \{1, ..., m\}$ be a mapping from indices of words in $\mathbf{q}$ to indices of words in $\mathbf{a}$. (It is for computational reasons that we assume $|x(i)| = 1$; in general x could range over *subsets* of $\{1, ..., m\}$.) Because we define the correspondence in this direction, note that it is possible for multple question words to map to the same answer word.

Why do we treat the alignment X as a hidden variable? In prior work, the alignment is assumed to be known given the sentences, but we aim to discover it from data. Our guide in this learning is the structure inherent in the QG: the configurations between parent-child pairs in the question and their corresponding, aligned words in the answer. The hidden variable treatment lets us avoid commitment to any one x mapping, making the method more robust to noisy parses (after all, the parser is not 100% accurate) and any wrong assumptions imposed by the model (that $|x(i)| = 1$, for example, or that syntactic transformations can explain the connection between $\mathbf{q}$ and $\mathbf{a}$ at all).[4]

Our model, then, defines

$$p(\mathbf{q}, \tau_{\mathbf{q}} \mid \mathbf{a}, \tau_{\mathbf{a}}) = \sum_x p(\mathbf{q}, \tau_{\mathbf{q}}, x \mid \mathbf{a}, \tau_{\mathbf{a}}) \tag{2}$$

where $\tau_{\mathbf{q}}$ and $\tau_{\mathbf{a}}$ are the question tree and answer tree, respectively. The stochastic process defined by our model factors cleanly into recursive steps that derive the question from the top down. The QG defines a grammar for this derivation; the grammar depends on the specific answer.

Let $\tau_{\mathbf{w}}^i$ refer to the subtree of $\tau_{\mathbf{w}}$ rooted at w_i. The model is defined by:

$$\begin{aligned}
& p(\tau_{\mathbf{q}}^i \mid q_i, \tau_{\mathbf{q}}(i), x(i), \tau_{\mathbf{a}}) = \qquad (3) \\
& \quad p_{\#kids}(|\{j : \tau_{\mathbf{q}}(j) = i, j < i\}| \mid q_i, \mathit{left}) \\
& \quad \times p_{\#kids}(|\{j : \tau_{\mathbf{q}}(j) = i, j > i\}| \mid q_i, \mathit{right}) \\
& \quad \times \prod_{j : \tau_{\mathbf{q}}(j) = i} \sum_{x(j)=0}^{m} \\
& \qquad\quad p_{kid}(q_j, \tau_{\mathbf{q}}(j).lab \mid q_i, \tau_{\mathbf{q}}(i), x(i), x(j), \tau_{\mathbf{a}}) \\
& \qquad\quad \times p(\tau_{\mathbf{q}}^j \mid q_j, \tau_{\mathbf{q}}(j), x(j), \tau_{\mathbf{a}})
\end{aligned}$$

[3] The main motivation for modeling $p(\mathbf{q} \mid \mathbf{a})$ is that it is easier to model *deletion* of information (such as the part of the sentence that answers the question) than *insertion*. Our QG does not model the real-world knowledge required to fill in an answer; its job is to know what answers are likely to look like, syntactically.

[4] If parsing performance is a concern, we might also treat the question and/or answer parse trees as hidden variables, though that makes training and testing more computationally expensive.

Note the recursion in the last line. While the above may be daunting, in practice it boils down only to defining the conditional distribution p_{kid}, since the number of left and right children of each node need not be modeled (the trees are assumed known)—$p_{\#kids}$ is included above for completeness, but in the model applied here we do not condition it on q_i and therefore do not need to estimate it (since the trees are fixed).

p_{kid} defines a distribution over syntactic children of q_i and their labels, given (1) the word q_i, (2) the parent of q_i, (3) the dependency relation between q_i and its parent, (4) the answer-word q_i is aligned to, (5) the answer-word the *child* being predicted is aligned to, and (6) the remainder of the answer tree.

4.2 Dynamic Programming

Given $\mathbf{q}$, the score for an answer is simply $p(\mathbf{q}, \tau_{\mathbf{q}} \mid \mathbf{a}, \tau_{\mathbf{a}})$. Computing the score requires summing over alignments and can be done efficiently by bottom-up dynamic programming. Let $S(j, \ell)$ refer to the score of $\tau_{\mathbf{q}}^j$, assuming that the parent of q_j, $\tau_{\mathbf{q}}(j).par$, is aligned to a_ℓ. The base case, for leaves of $\tau_{\mathbf{q}}$, is:

$$\begin{aligned} &S(j, \ell) = \qquad (4)\\ &\quad p_{\#kids}(0 \mid q_j, \mathit{left}) \times p_{\#kids}(0 \mid q_j, \mathit{right})\\ &\quad \times \sum_{k=0}^{m} p_{kid}(q_j, \tau_{\mathbf{q}}(j).lab \mid q_{\tau_{\mathbf{q}}(j)}, \ell, k, \tau_{\mathbf{a}}) \end{aligned}$$

Note that k ranges over indices of answer-words to be aligned to q_j. The recursive case is

$$\begin{aligned} &S(i, \ell) = \qquad (5)\\ &\quad p_{\#kids}(|\{j : \tau_{\mathbf{q}}(j) = i, j < i\}| \mid q_j, \mathit{left})\\ &\quad \times p_{\#kids}(|\{j : \tau_{\mathbf{q}}(j) = i, j > i\}| \mid q_j, \mathit{right})\\ &\quad \times \sum_{k=0}^{m} p_{kid}(q_i, \tau_{\mathbf{q}}(i).lab \mid q_{\tau_{\mathbf{q}}(i)}, \ell, k, \tau_{\mathbf{a}})\\ &\qquad \times \prod_{j : \tau_{\mathbf{q}}(j) = i} S(j, k) \end{aligned}$$

Solving these equations bottom-up can be done in $O(nm^2)$ time and $O(nm)$ space; in practice this is very efficient. In our experiments, computing the value of a question-answer pair took two seconds on average.[5] We turn next to the details of p_{kid}, the core of the model.

4.3 Base Model

Our base model factors p_{kid} into three conditional multinomial distributions.

$$\begin{aligned} &p_{kid}^{base}(q_i, \tau_{\mathbf{q}}(i).lab \mid q_{\tau_{\mathbf{q}}(i)}, \ell, k, \tau_{\mathbf{a}}) =\\ &\quad p(q_i.pos \mid a_k.pos) \times p(q_i.ne \mid a_k.ne)\\ &\quad \times p(\tau_{\mathbf{q}}(i).lab \mid \mathit{config}(\tau_{\mathbf{q}}, \tau_{\mathbf{a}}, i)) \qquad (6) \end{aligned}$$

where $q_i.pos$ is question-word i's POS label and $q_i.ne$ is its named-entity label. *config* maps question-word i, its parent, and their alignees to a QG configuration as described in Table 1; note that some configurations are extended with additional tree information. The base model does not directly predict the specific words in the question—only their parts-of-speech, named-entity labels, and dependency relation labels. This model is very similar to Smith and Eisner (2006).

Because we are interested in augmenting the QG with additional lexical-semantic knowledge, we also estimate p_{kid} by **mixing** the base model with a model that exploits WordNet (Miller et al., 1990) lexical-semantic relations. The mixture is given by:

$$p_{kid}(\bullet \mid \bullet) = \alpha p_{kid}^{base}(\bullet \mid \bullet) + (1-\alpha) p_{kid}^{ls}(\bullet \mid \bullet) \qquad (7)$$

4.4 Lexical-Semantics Log-Linear Model

The lexical-semantics model p_{kid}^{ls} is defined by predicting a (nonempty) subset of the thirteen classes for the question-side word given the identity of its aligned answer-side word. These classes include WordNet relations: identical-word, synonym, antonym (also extended and indirect antonym), hypernym, hyponym, derived form, morphological variation (e.g., plural form), verb group, entailment, entailed-by, see-also, and causal relation. In addition, to capture the special importance of Wh-words in questions, we add a special semantic relation called "q-word" between any word and any Wh-word. This is done through a log-linear model with one feature per relation. Multiple relations may fire, motivating the log-linear model, which permits "overlapping" features, and, therefore prediction of

[5] Experiments were run on a 64-bit machine with 2× 2.2GHz dual-core CPUs and 4GB of memory.

any of the possible $2^{13} - 1$ nonempty subsets. It is important to note that this model assigns zero probability to alignment of an answer-word with any question-word that is *not* directly related to it through *any* relation. Such words may be linked in the mixture model, however, via p_{kid}^{base}.[6]

(It is worth pointing out that log-linear models provide great flexibility in defining new features. It is straightforward to extend the feature set to include more domain-specific knowledge or other kinds of morphological, syntactic, or semantic information. Indeed, we explored some additional syntactic features, fleshing out the configurations in Table 1 in more detail, but did not see any interesting improvements.)

parent-child	Question parent-child pair align respectively to answer parent-child pair. Augmented with the q.-side dependency label.
child-parent	Question parent-child pair align respectively to answer child-parent pair. Augmented with the q.-side dependency label.
grandparent-child	Question parent-child pair align respectively to answer grandparent-child pair. Augmented with the q.-side dependency label.
same node	Question parent-child pair align to the same answer-word.
siblings	Question parent-child pair align to siblings in the answer. Augmented with the tree-distance between the a.-side siblings.
c-command	The parent of one answer-side word is an ancestor of the other answer-side word.
other	A catch-all for all other types of configurations, which are permitted.

Table 1: Syntactic alignment configurations are partitioned into these sets for prediction under the Jeopardy model.

4.5 Parameter Estimation

The parameters to be estimated for the Jeopardy model boil down to the conditional multinomial distributions in p_{kid}^{base}, the log-linear weights inside of p_{kid}^{ls}, and the mixture coefficient α.[7] Standard applications of log-linear models apply conditional maximum likelihood estimation, which for our case involves using an empirical distribution $\tilde{p}$ over question-answer pairs (and their trees) to optimize as follows:

$$\max_{\theta} \sum_{\mathbf{q},\tau_{\mathbf{q}},\mathbf{a},\tau_{\mathbf{a}}} \tilde{p}(\mathbf{q},\tau_{\mathbf{q}},\mathbf{a},\tau_{\mathbf{a}}) \log \underbrace{p_{\theta}(\mathbf{q},\tau_{\mathbf{q}} \mid \mathbf{a},\tau_{\mathbf{a}})}_{\sum_x p_{\theta}(\mathbf{q},\tau_{\mathbf{q}},x|\mathbf{a},\tau_{\mathbf{a}})} \tag{8}$$

Note the hidden variable x being summed out; that makes the optimization problem non-convex. This sort of problem can be solved in principle by conditional variants of the Expectation-Maximization algorithm (Baum et al., 1970; Dempster et al., 1977; Meng and Rubin, 1993; Jebara and Pentland, 1999). We use a quasi-Newton method known as L-BFGS (Liu and Nocedal, 1989) that makes use of the gradient of the above function (straightforward to compute, but omitted for space).

5 Experiments

To evaluate our model, we conducted experiments using Text REtrieval Conference (TREC) 8–13 QA dataset.[8]

5.1 Experimental Setup

The TREC dataset contains questions and answer patterns, as well as a pool of documents returned by participating teams. Our task is the same as Punyakanok et al. (2004) and Cui et al. (2005), where we search for single-sentence answers to factoid questions. We follow a similar setup to Shen and Klakow (2006) by automatically selecting answer candidate sentences and then comparing against a human-judged gold standard.

We used the questions in TREC 8–12 for training and set aside TREC 13 questions for development (84 questions) and testing (100 questions). To generate the candidate answer set for *development* and *testing*, we automatically selected sentences from each question's document pool that contains one or more non-stopwords from the question. For generating the training candidate set, in addtion to the sentences that contain non-stopwords from the question, we also added sentences that contain correct

[6] It is to preserve that robustness property that the models are *mixed*, and not combined some other way.

[7] In our experiments, all log-linear weights are initialized to be 1; all multinomial distributions are initialized as uniform distributions; α is initialized to be 0.1.

[8] We thank the organizers and NIST for making the dataset publicly available.

answer pattern. Manual judgement was produced for the entire TREC 13 set, and also for the first 100 questions from the training set TREC 8–12.[9] On average, each question in the development set has 3.1 positive and 17.1 negative answers. There are 3.6 positive and 20.0 negative answers per question in the test set.

We tokenized sentences using the standard treebank tokenization script, and then we performed part-of-speech tagging using MXPOST tagger (Ratnaparkhi, 1996). The resulting POS-tagged sentences were then parsed using MSTParser (McDonald et al., 2005), trained on the entire Penn Treebank to produce labeled dependency parse trees (we used a coarse dependency label set that includes twelve label types). We used BBN Identifinder (Bikel et al., 1999) for named-entity tagging.

As answers in our task are considered to be single sentences, our evaluation differs slightly from TREC, where an answer string (a word or phrase like *1977* or *George Bush*) has to be accompanied by a supporting document ID. As discussed by Punyakanok et al. (2004), the single-sentence assumption does not simplify the task, since the hardest part of answer finding is to locate the correct sentence. From an end-user's point of view, presenting the sentence that contains the answer is often more informative and evidential. Furthermore, although the judgement data in our case are more labor-intensive to obtain, we believe our evaluation method is a better indicator than the TREC evaluation for the quality of an answer selection algorithm.

To illustrate the point, consider the example question, "*When did James Dean die?*" The correct answer as appeared in the sentence "*In 1955, actor James Dean was killed in a two-car collision near Cholame, Calif.*" is *1955*. But from the same document, there is another sentence which also contains *1955*: "*In 1955, the studio asked him to become a technical adviser on Elia Kazan's 'East of Eden,' starring James Dean.*" If a system missed the first sentence but happened to have extracted *1955* from the second one, the TREC evaluation grants it a "correct and well-supported" point, since the document ID matches the correct document ID—even though the latter answer does not entail the true answer. Our evaluation does not suffer from this problem.

We report two standard evaluation measures commonly used in IR and QA research: mean average precision (MAP) and mean reciprocal rank (MRR). All results are produced using the standard `trec_eval` program.

5.2 Baseline Systems

We implemented two state-of-the-art answer-finding algorithms (Cui et al., 2005; Punyakanok et al., 2004) as strong baselines for comparison. Cui et al. (2005) is the answer-finding algorithm behind one of the best performing systems in TREC evaluations. It uses a mutual information-inspired score computed over dependency trees and a single alignment between them. We found the method to be brittle, often not finding a score for a testing instance because alignment was not possible. We extended the original algorithm, allowing fuzzy word alignments through WordNet expansion; both results are reported.

The second baseline is the approximate tree-matching work by Punyakanok et al. (2004). Their algorithm measures the similarity between $\tau_{\mathbf{q}}$ and $\tau_{\mathbf{a}}$ by computing tree edit distance. Our replication is close to the algorithm they describe, with one subtle difference. Punyakanok et al. used answer-typing in computing edit distance; this is not available in our dataset (and our method does not explicitly carry out answer-typing). Their heuristics for reformulating questions into statements were not replicated. We did, however, apply WordNet type-checking and approximate, penalized lexical matching. Both results are reported.

[9] More human-judged data are desirable, though we will address training from noisy, *automatically* judged data in Section 5.4. It is important to note that human judgement of answer sentence correctness was carried out prior to any experiments, and therefore is unbiased. The total number of questions in TREC 13 is 230. We exclude from the TREC 13 set questions that either have no correct answer candidates (27 questions), or no incorrect answer candidates (19 questions). Any algorithm will get the same performance on these questions, and therefore obscures the evaluation results. 6 such questions were also excluded from the 100 manually-judged training questions, resulting in 94 questions for training. For computational reasons (the cost of parsing), we also eliminated answer candidate sentences that are longer than 40 words from the training and evaluation set. After these data preparation steps, we have 348 positive Q-A pairs for training, 1,415 Q-A pairs in the development set, and 1,703 Q-A pairs in the test set.

		development set		test set	
training dataset	model	MAP	MRR	MAP	MRR
100 manually-judged	TreeMatch	0.4074	0.4458	0.3814	0.4462
	+WN	0.4328	0.4961	0.4189	0.4939
	Cui et al.	0.4715	0.6059	0.4350	0.5569
	+WN	0.5311	0.6162	0.4271	0.5259
	Jeopardy (base only)	0.5189	0.5788	0.4828	0.5571
	Jeopardy	**0.6812**	**0.7636**	**0.6029**	**0.6852**
+2,293 noisy	Cui et al.	0.2165	0.3690	0.2833	0.4248
	+WN	0.4333	0.5363	0.3811	0.4964
	Jeopardy (base only)	0.5174	0.5570	0.4922	0.5732
	Jeopardy	0.6683	0.7443	0.5655	0.6687

Table 2: Results on development and test sets. TreeMatch is our implementation of Punyakanok et al. (2004); +WN modifies their edit distance function using WordNet. We also report our implementation of Cui et al. (2005), along with our WordNet expansion (+WN). The Jeopardy base model and mixture with the lexical-semantics log-linear model perform best; both are trained using conditional maximum likelihood estimation. The top part of the table shows performance using 100 manually-annotated question examples (questions 1–100 in TREC 8–12), and the bottom part adds noisily, automatically annotated questions 101–2,393. Boldface marks the best score in a column and any scores in that column not significantly worse under a a two-tailed paired t-test ($p < 0.03$).

5.3 Results

Evaluation results on the development and test sets of our model in comparison with the baseline algorithms are shown in Table 2. Both our model and the model in Cui et al. (2005) are trained on the manually-judged training set (questions 1-100 from TREC 8–12). The approximate tree matching algorithm in Punyakanok et al. (2004) uses fixed edit distance functions and therefore does not require training. From the table we can see that our model significantly outperforms the two baseline algorithms—even when they are given the benefit of WordNet—on both development and test set, and on both MRR and MAP.

5.4 Experiments with Noisy Training Data

Although manual annotation of the remaining 2,293 training sentences' answers in TREC 8–12 was too labor-intensive, we did experiment with a simple, noisy automatic labeling technique. Any answer that had at least three non-stop word types seen in the question *and* contains the answer pattern defined in the dataset was labeled as "correct" and used in training. The bottom part of Table 2 shows the results. Adding the noisy data hurts all methods, but the Jeopardy model maintains its lead and consistently suffers less damage than Cui et al. (2005). (The TreeMatch method of Punyakanok et al. (2004) does not use training examples.)

5.5 Summing vs. Maximizing

Unlike most previous work, our model does not try to find a single correspondence between words in the question and words in the answer, during training or during testing. An alternative method might choose the *best* (most probable) alignment, rather than the sum of all alignment scores. This involves a slight change to Equation 3, replacing the summation with a maximization. The change could be made during training, during testing, or both. Table 3 shows that summing is preferable, especially during training.

6 Discussion

The key experimental result of this work is that loose syntactic transformations are an effective way to carry out statistical question answering.

One unique advantage of our model is the mixture of a factored, multinomial-based base model and a potentially very rich log-linear model. The base model gives our model robustness, and the log-

		test set	
training	decoding	MAP	MRR
Σ	Σ	**0.6029**	**0.6852**
Σ	$\max$	0.5822	0.6489
$\max$	Σ	0.5559	0.6250
$\max$	$\max$	0.5571	0.6365

Table 3: Experimental results on comparing summing over alignments (Σ) with maximizing ($\max$) over alignments on the test set. Boldface marks the best score in a column and any scores in that column not significantly worse under a a two-tailed paired t-test ($p < 0.03$).

linear model allows us to throw in task- or domain-specific features. Using a mixture gives the advantage of smoothing (in the base model) without having to normalize the log-linear model by summing over large sets. This powerful combination leads us to believe that our model can be easily ported to other semantic processing tasks where modeling syntactic and semantic transformations is the key, such as textual entailment, paraphrasing, and cross-lingual QA.

The traditional approach to cross-lingual QA is that translation is either a pre-processing or post-processing step done independently from the main QA task. Notice that the QG formalism that we have employed in this work was originally proposed for machine translation. We might envision transformations that are performed together to form questions from answers (or vice versa) *and* to translate—a *Jeopardy!* game in which bilingual players must ask a question in a different language than that in which the answer is posed.

7 Conclusion

We described a statistical syntax-based model that softly aligns a question sentence with a candidate answer sentence and returns a score. Discriminative training and a relatively straightforward, barely-engineered feature set were used in the implementation. Our scoring model was found to greatly outperform two state-of-the-art baselines on an answer selection task using the TREC dataset.

Acknowledgments

The authors acknowledge helpful input from three anonymous reviewers, Kevin Gimpel, and David Smith. This work is supported in part by ARDA/DTO Advanced Question Answering for Intelligence (AQUAINT) program award number NBCHC040164.

References

Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi, Alessandro Tommasi, Ellen M. Voorhees, and D. K. Harman. 2001. Selectively using relations to improve precision in question answering. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, Gaithersburg, MD, USA.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns whatś in a name. *Machine Learning*, 34(1-3):211–231.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th ACM-SIGIR International Conference on Research and Development in Information Retrieval*, Salvador, Brazil.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43st Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, MI, USA.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44st Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia.

Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia.

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system—TREC-10. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, Gaithersburg, MD, USA.

Tony Jebara and Alex Pentland. 1999. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II (NIPS)*, pages 494–500, Denver, CO, USA.

Boris Katz and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*, Gaithersburg, MD, USA.

Jinxi Xu Ana Licuanan and Ralph Weischedel. 2003. Trec2003 qa at bbn: Answering definitional questions. In *Proceedings of the 12th Text REtrieval Conference (TREC-12)*, Gaithersburg, MD, USA.

Kenneth C. Litkowski. 1999. Question-answering using semantic relation triples. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, USA.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, New York, NY, USA.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernado Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43st Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, MI, USA.

I. Dan Melamed. 2004. Algorithms for syntax-aware statistical machine translation. In *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Baltimore, MD, USA.

Xiao-Li Meng and Donald B. Rubin. 1993. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80:267–278.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4).

Vanessa Murdock and W. Bruce Croft. 2005. A translation model for sentence retrieval. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, Vancouver, BC, USA.

Vasin Punyakanok, Dan Roth, and Wen-Tau Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, FL, USA.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, Ann Arbor, MI, USA.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, PA, USA.

Deepak Ravichandran, Abharam Ittycheriah, and Salim Roukos. 2003. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia.

Dan Shen, Geert-Jan M. Kruijff, and Dietrich Klakow. 2005. Exploring syntactic relation patterns for question answering. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP)*, Jeju Island, Republic of Korea.

Hideki Shima, Mengqiu Wang, Frank Lin, and Teruko Mitamura. 2006. Modular approach to error analysis and evaluation for multilingual question answering. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, New York, NY, USA.

Martin M. Soubbotin and Sergei M. Soubbotin. 2001. Patterns for potential answer expressions as clues to the right answers. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, Gaithersburg, MD, USA.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING)*, Montreal, Canada.

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, MI, USA.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, Toulouse, France.

Learning Unsupervised SVM Classifier for Answer Selection in Web Question Answering

Youzheng Wu, Ruiqiang Zhang, Xinhui Hu, and Hideki Kashioka
National Institute of Information and Communications Technology (NICT),
ATR Spoken Language Communication Research Labs.
2-2-2 Hikaridai ”Keihanna Science City” Kyoto 619-0288 Japan
{Youzheng.wu,Ruiqiang.zhang,Xinhui.hu,Hideki.kashioka}@atr.jp

Abstract

Previous machine learning techniques for answer selection in question answering (QA) have required question-answer training pairs. It has been too expensive and labor-intensive, however, to collect these training pairs. This paper presents a novel unsupervised support vector machine (U-SVM) classifier for answer selection, which is independent of language and does not require hand-tagged training pairs. The key ideas are the following: 1. unsupervised learning of training data for the classifier by clustering web search results; and 2. selecting the correct answer from the candidates by classifying the question. The comparative experiments demonstrate that the proposed approach significantly outperforms the retrieval-based model (Retrieval-M), the supervised SVM classifier (S-SVM), and the pattern-based model (Pattern-M) for answer selection. Moreover, the cross-model comparison showed that the performance ranking of these models was: U-SVM > Pattern-M > S-SVM > Retrieval-M.

1 Introduction

The purpose of answer selection in QA is to select the exact answer to the question from the extracted candidate answers. In recent years, many supervised machine learning techniques for answer selection in open-domain question answering have been investigated in some pioneering studies [Ittycheriah et al. 2001; Ng et al. 2001; Suzuki et al. 2002; Sasaki, et al. 2005; and Echihabi et al. 2003]. Compared with retrieval-based [Yang et al. 2003], pattern-based [Ravichandran et al. 2002 and Soubbotin et al. 2002], and deep NLP-based [Moldovan et al. 2002, Hovy et al. 2001; and Pasca et al. 2001] answer selection, machine learning techniques are more effective in constructing QA components from scratch. These techniques suffer, however, from the problem of requiring an adequate number of hand-tagged question-answer training pairs. It is too expensive and labor intensive to collect such training pairs for supervised machine learning techniques.

To tackle this knowledge acquisition bottleneck, this paper presents an unsupervised SVM classifier for answer selection, which is independent of language and question type, and avoids the need for hand-tagged question-answer pairs. The key ideas are as follows:

1. Regarding answer selection as a kind of classification task and adopting an SVM classifier;

2. Applying unsupervised learning of pseudo-training data for the SVM classifier by clustering web search results;

3. Training the SVM classifier by using three types of features extracted from the pseudo-training data; and

4. Selecting the correct answer from the candidate answers by classifying the question. Note that this means classifying a question into one of the clusters learned by clustering web search results. Therefore, our classifying the question

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 33–41, Prague, June 2007. ©2007 Association for Computational Linguistics

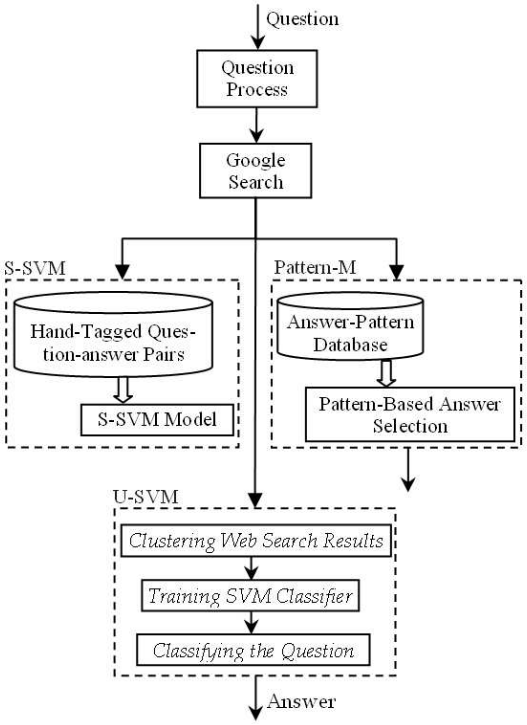


Figure 1: Web Question Answering Architecture

> is different from conventional question classification (QC) [Li et al. 2002] that determines the answer type of the question.

The proposed approach is fully unsupervised and starts only from a user question. It does not require richly annotated corpora or any deep linguistic tools. To the best of our knowledge, no research on this kind of study we discuss here has been reported. Figure 1 illustrates the architecture of our web QA approach. The S-SVM and Pattern-M models are included for comparison.

Because the focus of this paper just evaluates the answer selection part, our approach requires knowledge of the answer type to the question in order to find candidate answers, and that the answer must be a NE for convenience in candidate extraction. Experiments using Chinese versions of the TREC 2004 and 2005 test data sets show that our approach significantly outperforms the S-SVM for answer selection, with a top_1 score improvement of more than 20%. Results obtained with the test data set in [Wu et al. 2004] show that the U-SVM increases the $top\_1/mrr\_5/top\_5$ scores by 5.95%/6.06%/8.68% as compared with the Pattern-M. Moreover, our cross-model comparison demonstrates that the performance ranking of all models considered is: U-SVM $>$ Pattern-M $>$ S-SVM $>$ Retrieval-M.

2 Comparison among Models

Related researches on answer selection in QA can be classified into four categories. The retrieval-based model [Yang et al. 2003] selects a correct answer from the candidates according to the distance between a candidate and all question keywords. This model does not work, however, if the question and the answer-bearing sentences do not match on the surface. The pattern-based model [Ravichandran et al. 2002 and Soubbotin et al. 2002] first classifies the question into predefined categories, and then extracts the exact answer by using answer patterns learned off-line. Although the pattern-based model can obtain high precision for some predefined types of questions, it is difficult to define question types in advance for open-domain question answering. Furthermore, this model is not suitable for all types of questions. The deep NLP-based model [Moldovan et al. 2002; Hovy et al. 2001; and Pasca et al. 2001] usually parses the user question and an answer-bearing sentence into a semantic representation, and then semantically matches them to find the answer. This model has performed very well at TREC workshops, but it heavily depends on high-performance NLP tools, which are time consuming and labor intensive for many languages. Finally, the machine learning-based model has also been investigated. current models of this type are based on supervised approaches [Ittycheriah et al. 2001; Ng et al. 2001; Suzuki et al. 2002; and Sasaki et al. 2005] that are heavily dependent on hand-tagged question-answer training pairs, which not readily available.

In response to this situation, this paper presents the U-SVM for answer selection in open-domain web question answering system. Our U-SVM has the following advantages over supervised machine learning techniques. First, the U-SVM classifies questions into a question-dependent set of clusters, and the answer is the name of a question cluster. In contrast, most previous models have classified candidates into positive and negative. Second, the U-SVM automatically learns the unique question-dependent clusters and the pseudo-training for each

Table 1: Comparison of Various Machine Learning Techniques

System	Model	Key Idea	Training Data
[Ittycheriah et al. 2001]	ME Classifier	Classifying candidates into positive and negative	5,000 English Q-A pairs
[Suzuki et al. 2002]	SVM Classifier	Classifying candidates into positive and negative	1358 Japanese Q-A pairs
[Echihabi et al. 2003]	N-C Model	Selecting correct answer by aligning question with sentences	90,000 English Q-A pairs
[Sasaki et al. 2005]	ME Classifier	Classifying words in sentences into answer and non-answer words	2,000 Japanese Q-A pairs
Our U-SVM Model	SVM Classifier	Classifying question into a set of question-dependent clusters	No Q-A pairs

question. This differs from the supervised techniques, in which a large number of hand-tagged training pairs are shared by all of the test questions. In addition, supervised techniques independently process the answer-bearing sentences, so the answers to the questions may not always be extractable because of algorithmic limitations. On the other hand, the U-SVM can use the interdependence between answer-bearing sentences to select the answer to a question.

Table 1 compares the key idea and training data used in the U-SVM with those used in the supervised machine learning techniques. Here, ME means the maximum entropy model, and N-C means the noisy-channel model.

3 The U-SVM

The essence of the U-SVM is to regard answer selection as a kind of text categorization-like classification task, but with no training data available. In the U-SVM, the steps of ”clustering web search results”, ”classifying the question”, and ”training SVM classifier” play very important roles.

3.1 Clustering Web Search Results

Web search results, such as snippets returned by Google, usually include a mixture of multiple subtopics (called clusters in this paper) related to the user question. To group the web search results into clusters, we assume that the candidate answer in each Google snippet can represent the ”signature” of its cluster. In other words, the Google snippets containing the same candidate are regarded as aligned snippets, and thus belong to the same cluster. Web search results are clustered in two phases.

- A first-stage Google search (FGS) is applied to extract n candidate answers $\{c_1, c_2, \ldots, c_n\}$ from the top m Google snippets $\{s_1, s_2, \ldots, s_m\}$ by a NER tool [Wu et al. 2005]. Those snippets containing the candidates $\{c_i\}$ and at least one question keyword $\{q_i\}$ are retained. Finally, the retained snippets $\{s_1, s_2, \ldots, s_m\}$ are clustered into n clusters $\{C_1, C_2, \ldots, C_n\}$ by clustering web search results, that is,

  If a snippet includes L different candidates, the snippet belongs to L different clusters.
  If the candidates of different snippets are the same, these snippets belong to the same clusters.

  Consequently, the number of clusters $\{C_i\}$ is fully determined by the number of candidates $\{c_i\}$, and the cluster name of a cluster C_i is the candidate answer c_i. Up to this point, we have obtained clusters and sample snippets for each cluster that will be used as training data for the SVM classifier. Because this training data is learned automatically, rather than hand-tagged, we call it pseudo-training data.

- A second-stage Google search (SGS) is applied to resolve data sparseness in the pseudo-training samples learned through the FGS. The FGS data may have very few training snippets in some clusters, so more snippets must be collected. Note that this step just learns new

Google snippets into the clusters learned by the FGS, but does not add new clusters.

```
For each candidate answer $c_i$:
    Combine the original query $q = \{q_i\}$ and
    the candidate $c_i$ to form a new query $q\prime =$
    $\{q, c_i\}$.
    Submit $q\prime$ to Google and download the top
    50 Google snippets.
    Retain the snippets containing the candi-
    date $c_i$ and at least one keyword $q_i$.
    Group the retained snippets into $n$ clusters
    to form the new pseudo-training data.
End
```

Here, we give an example illustrating the principle of clustering web search results in the FGS. In submitting TREC 2004 test question 1.1 ”when was the first Crip gang started?” to Google (*http://www.google.com/apis*), we extract $n(= 8)$ different candidates from the top $m(= 30)$ Google snippets. The Google snippets containing the same candidates are aligned snippets, and thus the 12 retained snippets are grouped into 8 clusters, as listed in Table 2. This table roughly indicates that the snippets with the same candidate answers contain the same sub-meanings, so these snippets are considered as aligned snippets. For example, all Google snippets that contain the candidate answer *1969* express the time of establishment of ”the first Crip gang”.

In summary, the U-SVM uses the result of ”clustering web search results” as the pseudo-training data of the SVM classifier, and then classifies user question into one of the clusters for answer selection. On the one hand, the clusters and their names are based on candidate answers to question; on the other hand, candidates are dependent on question. Therefore, the clusters are question-dependent.

3.2 Classifying Question

Using the pseudo-training data obtained by clustering web search results to train the SVM classifier, we classify user questions into a set of question-dependent clusters and assume that the correct answer is the name of the question cluster that is assigned by the trained U-SVM classifier. For the above example, if the U-SVM classifier, trained on the pseudo-training data listed in Table 2, classifies the above test question into a cluster whose name is *1969*, then the cluster name *1969* is the answer to the question.

This paper selects LIBSVM toolkit[1] to implement the SVM classifier. The kernel is the radical basis function with the parameter $\gamma = 0.001$ in the experiments.

3.3 Feature Extraction

To classify the question into a question-dependent set of clusters, the U-SVM classifier extracts three types of features.

- A similarity-based feature set (SBFS) is extracted from the Google snippets. The SBFS attempts to capture the word overlap between a question and a snippet. The possible values range from 0 to 1.

SBFS Features
percentage of matched keywords (KWs)
percentage of mismatched KWs
percentage of matched bi-grams of KWs
percentage of matched thesauruses
normalized distance between candidate and KWs

To compute the matched thesaurus feature, we adopt TONGYICICILIN [2] in the experiments.

- A Boolean match-based feature set (BMFS) is also extracted from the Google snippets. The BMFS attempts to capture the specific keyword Boolean matches between a question and a snippet. The possible values are true or false.

BMFS Features
person names are matched or not
location names are matched or not
organization names are matched or not
time words are matched or not
number words are matched or not
root verb is matched or not
candidate has or does not have bi-gram in snippet matching bi-gram in question
candidate has or does not have desired named entity type

- A window-based word feature set (WWFS) is a set of words consisting of the words

[1] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[2] A Chinese Thesaurus Lexicon

Table 2: Clustering Web Search Results

Cluster Name	Google Snippet
1969	It is believed that the first Crip gang was formed in late 1969. During this time in Los Angeles there were ...
	... the first Bloods and Crips gangs started forming in Los Angeles in late 1969, the Island Bloods sprung up in north Pomona ...
	... formed by 16 year old Raymond Lee Washington in 1969. Williams joined Washington in 1971 ... had come to be called the Crips. It was initially started to eliminate all street gangs ...
August 8, 2005	High Country News – August 8, 2005: The Gangs of Zion
2004	2004 main 1 Crips 1.1 FACTOID When was the first Crip gang started? 1.2 FACTOID What does the name mean or come...
1972	One of the first-known and publicized killings by Crip gang members occurred at the Hollywood Bowl in March 1972.
1971	Williams joined Washington in 1971, forming the westside faction of what had come to be called the Crips.
	The Crips gang formed as a kind of community watchdog group in 1971 after the demise of the Black Panthers. ...
	... formed by 16 year old Raymond Lee Washington in 1969. Williams joined Washington in 1971 ... had come to be called the Crips. It was initially started to eliminate all street gangs ...
1982	Oceanside police first started documenting gangs in 1982, when five known gangs were operating in the city: the Posole Locos...
mid-1990s	Street Locos; Deep Valley Bloods and Deep Valley Crips. By the mid-1990s, gang violence had ...
1970s	The Blood gangs started up as opposition to the Crips gangs, also in the 1970s, and the rivalry stands to this day ...

preceding $\{w_{i-5}, \ldots, w_{i-1}\}$ and following $\{w_{i+1}, \ldots, w_{i+5}\}$ the candidate answer. The WWFS features can be regarded as a kind of relevant snippets-based question keywords expansion. By extracting the WWFS feature set, the feature space in the U-SVM becomes question dependent, which may be more suitable for classifying the question. The number of classification features in the S-SVM must be fixed, however, because all questions share the same training data. This is one difference between the U-SVM and the supervised SVM classifier for answer selection. Each word feature in the WWFS is weighted using its ISF value.

$$ISF(w_j, C_i) = \frac{N(w_j, C_i) + 0.5}{N(w_j) + 0.5} \qquad (1)$$

where $N(w_j)$ is the total number of the snippets containing word feature w_j, and $N(w_j, C_i)$ is the number of snippets in cluster C_i containing word feature w_j.

When constructing question vector, we assume that the question is an ideal question that contains all the extracted WWFS words. Therefore, the values of the WWFS word features in question vector are 1. Similarly, the values of the SBFS and BMFS features in question vector are also estimated by self-similarity calculation.

4 Experiments

4.1 Data Sets

For the experiments, no English named entity recognition (NER) tool is in our hand at the time of the experiments; therefore, we validate the U-SVM in terms of Chinese web QA using three test data sets, which will be published with this paper[3]. Although the U-SVM is independent of the question types, for convenience in candidate extraction, only those questions whose answers are named entities are selected. The three test data sets are CTREC04, CTREC05 and CTEST05. CTREC04 is a set of 178 Chinese questions translated from TREC 2004 FACTOID testing questions. CTREC05 is a set of 279 Chinese questions translated from TREC 2005 FACTOID testing questions. CTEST05 is a set of 178 Chinese questions found in [Wu et al. 2004] that are similar to TREC testing questions except that they are written in Chinese. Figure 2 breaks down the types of questions (manually assigned) in the CTREC04 and CTREC05 data sets. Here, PER, LOC, ORG, TIM, NUM, and CR refer to questions whose answers are a person, location, organization, time, number, and book or movie, respectively.

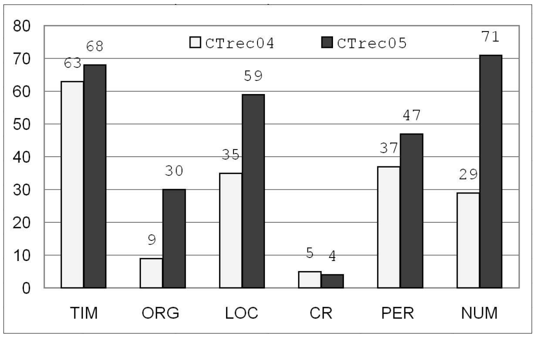


Figure 2: Statistics of CTEST05

To collect the question-answer training data for the S-SVM, we submitted 807 Chinese questions to Google and extracted the candidates for each question from the top *50* Google snippets. We then manually selected the snippets containing the correct answers as positive snippets, and designated all of the other snippets as negative snippets. Finally, we collected 807 hand-tagged Chinese question-answer pairs as the training data of S-SVM called CTRAINDATA.

4.2 Evaluation Method

In the experiments, the top $m(=50)$ Google snippets are adopted to extract candidates by using a Chinese NER tool [Wu et al. 2005]. The number of the candidates extracted from the top $m(=50)$ snippets, n, is adaptive for different questions but it does not exceed 30. The results are evaluated in terms of two scores, top_n and mrr_5. Here, top_n is the rate at which at least one correct answer is included in the top n answers, while mrr_5 is the average reciprocal rank $(1/n)$ of the highest rank $n(n \leq 5)$ of a correct answer to each question.

4.3 U-SVM vs. Retrieval-M

The Retrieval-M selects the candidate with the shortest distances to all question keywords as the correct answer. In this experiment, the Retrieval-M is implemented based on the snippets returned by Google, while the U-SVM is based on the SGS data, the SBFS and BMFS feature. Table 3 summarizes the comparative performance.

Table 3: Comparison of Retrieval-M and U-SVM

		Retrieval-M	U-SVM
CTREC04	top_1	27.84%	53.61%
	mrr_5	43.67%	66.25%
	top_5	71.13%	88.66%
CTREC05	top_1	34.00%	50.00%
	mrr_5	48.20%	62.38%
	top_5	71.33%	82.67%

The table shows that the U-SVM greatly improves the performance of the Retrieval-M: the top_1 improvements for CTREC04 and CTREC05 are about 25.8% and 16.0%, respectively. This experiment demonstrates that the assumptions used here in clustering web search results and in classifying the question are effective in many cases, and that the U-SVM benefits from these assumptions.

4.4 U-SVM vs. S-SVM

To explore the effectiveness of our unsupervised model as compared with the supervised model, we conduct a cross-model comparison of the S-SVM and the U-SVM with the SBFS and BMFS feature sets. The U-SVM results are compared with the S-SVM results for CTREC04 and CTREC05 in Tables 4 and 5, respectively. The S-SVM is trained on CTRAINDATA.

These tables show the following:

[3] Currently no public testing question set for simplified Chinese QA is available.

Table 4: Comparison of U-SVM and S-SVM on CTREC04

		FGS	SGS
top_1	S-SVM	30.93%	39.18%
	U-SVM	45.36%	53.61%
mrr_1	S-SVM	45.36%	53.54%
	U-SVM	57.44%	66.25%
top_5	S-SVM	71.13%	79.38%
	U-SVM	76.29%	88.66%

Table 5: Comparison of U-SVM and S-SVM on CTREC05

		FGS	SGS
top_1	S-SVM	30.00%	33.33%
	U-SVM	48.00%	50.00%
mrr_1	S-SVM	45.59%	48.67%
	U-SVM	58.01%	62.38%
top_5	S-SVM	72.00%	74.67%
	U-SVM	75.33%	82.67%

- The proposed U-SVM significantly outperforms the S-SVM for all measurements and all test data sets. For the CTREC04 test data set, the top_1 improvements for the FGS and SGS data are about 14.5% and 14.4%, respectively. For the CTREC05 test data set, the top_1 score for the FGS data increases from 30.0% to 48.0%, and the top_1 score for the SGS data increases from 33.3% to 50.0%. Note that the SBFS and BMFS features here is fewer than the features in [Ittycheriah et al. 2001; Suzuki et al. 2002], but the comparison is still effective because the models are compared in terms of the same features. In the S-SVM, all questions share the same training data, while the U-SVM uses the unique pseudo-training data for each question. This is the main reason why the U-SVM performs better than the S-SVM does.

- The SGS data is greatly helpful for both the U-SVM and the S-SVM. Compared with the FGS data, the $top\_1/mrr\_5/top\_5$ improvements for the S-SVM and the U-SVM on CTREC04 are 8.25%/8.18%/8.25% and 7.25%/8.81%/12.37%. The SGS can be regarded as a kind of query expansion. The reasons for this improvement are: the data sparseness in FGS data is partially resolved; and the use of the Web to introduce data redundancy is helpful. [Clarke et al. 2001; Magnini et al. 2002; and Dumais et al. 2002].

In the S-SVM, all of the test questions share the same hand-tagged training data, so the WWFS features cannot be easily used [Ittycheriah et al. 2002; Suzuki, et al. 2002]. Tables 6 and 7 compare the performances of the U-SVM with the (SBFS + BMFS) features, the WWFS features, and combination of three types of features for the CTREC04 and CTREC05 test data sets, respectively.

Table 6: Performances of U-SVM for Different Features on CTREC04

	SBFS+BMFS	WWFS	Combination
top_1	53.61%	46.39%	60.82%
mrr_5	66.25%	59.19%	71.31%
top_5	88.66%	81.44%	88.66%

Table 7: Performances of U-SVM for Different Features on CTREC05

	SBFS+BMFS	WWFS	Combination
top_1	50.00%	49.33%	57.33%
mrr_5	62.38%	59.26%	65.61%
top_5	82.67%	74.00%	80.00%

These tables report that combining three types of features can improve the performance of the U-SVM. Using a combination of features with the CTREC04 test data set results in the best performances: 60.82%/71.31%/88.66% for $top\_1/mrr\_5/top\_5$. Similarly, as compared with using the (SBFS + BMFS) and WWFS features, the improvements from using a combination of features with the CTREC05 test data set are 7.33%/3.23%/-2.67% and 8.00%/6.35%/6.00%, respectively. The results also demonstrate that the (SBFS + BMFS) features are more important than the WWFS features.

These comparative experiments indicate that the U-SVM performs better than the S-SVM does, even though the U-SVM is an unsupervised technique and no hand-tagged training data is provided. The average top_1 improvements for both test data sets are both more than 20%.

4.5 U-SVM vs. Pattern-M vs. S-SVM

To compare the U-SVM with the Pattern-M and the S-SVM, we use the CTEST05 data set, shown in Figure 3. The CTEST05 includes 14 different question types, for example, Inventor_Stuff (with question like "Who invented telephone?"), Event-Day (with question like "when is World Day for Water?"), and so on. The Pattern-M uses the dependency syntactic answer patterns learned in [Wu et al. 2007] to extract the answer, and named entities are also used to filter noise from the candidates.

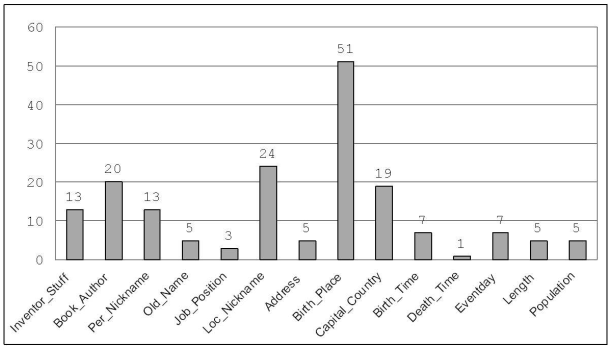


Figure 3: Statistics of CTEST05

Table 8 summarizes the performances of the U-SVM, Pattern-M, and S-SVM models on CTEST05.

Table 8: Comparison of U-SVM, Pattern-M and S-SVM on CTEST05

	S-SVM	Pattern-M	U-SVM
top_1	44.89%	53.14%	59.09%
mrr_5	56.49%	61.28%	67.34%
top_5	74.43%	73.14%	81.82%

The results in the table show that the U-SVM significantly outperforms the S-SVM and Pattern-M, while the S-SVM underperforms the Pattern-M. Compared with the Pattern-M, the U-SVM increases the $top\_1/mrr\_5/top\_5$ scores by 5.95%/ 6.06%/8.68%, respectively. The reasons may lie in the following:

- The Chinese dependency parser influences dependency syntactic answer-pattern extraction, and thus degrades the performance of the Pattern-M model.
- The imperfection of Google snippets affects pattern matching, and thus adversely influences the Pattern-M model. From the cross-model comparison, we conclude that the performance ranking of these models is: U-SVM $>$ Pattern-M $>$ S-SVM $>$ Retrieval-M.

5 Conclusion and Future Work

This paper presents an unsupervised machine learning technique (called the U-SVM) for answer selection that is validated in Chinese open-domain web QA. Regarding answer selection as a kind of classification task, the U-SVM automatically learns clusters and pseudo-training data for each cluster by clustering web search results. It then selects the correct answer from the candidates according to classifying the question. The contribution of this paper is that it presents an unsupervised machine learning technique for web QA that starts with only a user question. The results of our experiments with three test data sets are encouraging. As compared with the S-SVM, the top_1 performances of the U-SVM for the CTREC04 and CTREC05 data sets are significantly improved, at more than 20%. Moreover, the U-SVM performs better than the Retrieval-M and the Pattern-M.

These experiments have only validated the U-SVM on named entity types of questions that account for about 82% of all TREC2004 and 2005 FACTOID test questions. In fact, our technique is independent of question types only if the candidates can be extracted. In the future, we will explore the effectiveness of our technique for the other types of questions. The web search results clustering in the U-SVM defines that a candidate in a Google snippet can represent the "signature" of its cluster. This definition, however, is not always effective. To filter noise in the pseudo-training data, we will extract relations between the candidates and the keywords as the cluster signatures of Google snippets. Moreover, applying the U-SVM to QA systems in other languages, like English and Japanese, will also be included in our future work.

References

Abdessamad Echihabi, and Daniel Marcu. 2003. *A Noisy-Channel Approach to Question Answering*. In Proc. of ACL-2003, Japan.

Abraham Ittycheriah, Salim Roukos. 2002. *IBM's Statistical Question Answering System-TREC 11*. In Proc. of TREC-11, Gaithersburg, Maryland.

Bernardo Magnini, Matteo Negri, Roberto Prevete, Hristo Tanev. 2002. *Is It the Right Answer? Exploiting Web Redundancy for Answer Validation*. In Proc. of ACL-2002, Philadelphia, pp. 425 432.

Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam. *Exploiting Redundancy in Question Answering* In Proc. of SIGIR-2001, pp 358–365, 2001.

Christopher Pinchak, Dekang Lin. 2006. *A Probabilistic Answer Type Model*. In Proc. of EACL-2006, Trento, Italy, pp. 393-400.

Dan Moldovan, Sanda Harabagiu, Roxana Girju, et al. 2002. *LCC Tools for Question Answering*. NIST Special Publication: SP 500-251, TREC-2002.

Deepak Ravichandran, Eduard Hovy. 2002. *Learning Surface Text Patterns for a Question Answering System*. In Proc. of the 40th ACL, Philadelphia, July 2002.

Eduard Hovy, Ulf Hermjakob, Chin-Yew Lin. 2001. *The Use of External Knowledge of Factoid QA*. In Proc. of TREC 2001, Gaithersburg, MD, U.S.A., November 13-16, 2001.

Hui Yang, Tat-Seng Chua. 2003. *QUALIFIER: Question Answering by Lexical Fabric and External Resources*. In Proc. of EACL-2003, page 363-370.

Hwee T. Ng, Jennifer L. P. Kwan, and Yiyuan Xia. 2001. *Question Answering Using a Large Text Database: A Machine Learning Approach*. In Proc. of EMNLP-2001, pp66-73 (2001).

Jun Suzuki, Yutaka Sasaki, Eisaku Maeda. 2002. *SVM Answer Selection for Open-Domain Question Answering*. In Proc. of Coling-2002, pp. 974 980 (2002).

Marius Pasca. 2001. *A Relational and Logic Representation for Open-Domain Textual Question Answering*. In Proc. of ACL (Companion Volume) 2001: 37-42.

Martin M. Soubbotin, Sergei M. Soubbotin. 2002. *Use of Patterns for Detection of Likely Answer Strings: A Systematic Approach*. In Proc. of TREC-2002, Gaithersburg, Maryland, November 2002.

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andre Ng. *Web Question Answering: Is More Always Better?*. In Proc. SIGIR-2002, pp 291–298, 2002.

Xin Li, and Dan Roth. 2002. *Learning Question Classification*. In Proc. of the 19th International Conference on Computational Linguistics, Taibai, 2002.

Youzheng Wu, Hideki Kashioka, Jun Zhao. 2007. *Using Clustering Approaches to Open-domain Question Answering*. In Proc. of CICLING-2007, Mexico City, Mexico, pp506 517, 2007.

Youzheng Wu, Jun Zhao and Bo Xu. 2005. *Chinese Named Entity Recognition Model Based on Multiple Features*. In Proc. of HLT/EMNLP-2005, Vancouver, Canada, pp.427-434.

Youzheng Wu, Jun Zhao, Xiangyu Duan and Bo Xu. 2004. *Building an Evaluation Platform for Chinese Question Answering Systems*. In Proc. of the First NCIRCS, China, December, 2004.

Yutaka Sasaki. 2005. *Question Answering as Question-Biased Term Extraction: A New Approach toward Multilingual QA*. In Proc. of ACL-2005, pp.215-222.

Improving Word Alignment with Bridge Languages

Shankar Kumar and **Franz Och** and **Wolfgang Macherey**
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, U.S.A.
{shankarkumar,och,wmach}@google.com

Abstract

We describe an approach to improve Statistical Machine Translation (SMT) performance using multi-lingual, parallel, sentence-aligned corpora in several bridge languages. Our approach consists of a simple method for utilizing a bridge language to create a word alignment system and a procedure for combining word alignment systems from multiple bridge languages. The final translation is obtained by consensus decoding that combines hypotheses obtained using all bridge language word alignments. We present experiments showing that multilingual, parallel text in Spanish, French, Russian, and Chinese can be utilized in this framework to improve translation performance on an Arabic-to-English task.

1 Introduction

Word Alignment of parallel texts forms a crucial component of phrase-based statistical machine translation systems. High quality word alignments can yield more accurate phrase-pairs which improve quality of a phrase-based SMT system (Och and Ney, 2003; Fraser and Marcu, 2006b).

Much of the recent work in word alignment has focussed on improving the word alignment quality through better modeling (Och and Ney, 2003; Deng and Byrne, 2005; Martin et al., 2005) or alternative approaches to training (Fraser and Marcu, 2006b; Moore, 2005; Ittycheriah and Roukos, 2005). In this paper we explore a complementary approach to improve word alignments using multi-lingual, parallel (or multi-parallel) corpora. Two works in the literature are very relevant to our approach. Borin (2000) describes a non-statistical approach where a pivot alignment is used to combine direct translation and indirect translation via a third language. Filali and Bilmes (2005) present a multi-lingual extension to the IBM/HMM models. Our current approach differs from this latter work in that we propose a simple framework to combine word alignments from any underlying statistical alignment model without the need for changing the structure of the model. While both of the above papers focus on improving word alignment quality, we demonstrate that our approach can yield improvements in translation performance. In particular, we aim to improve an Arabic-to-English (Ar-En) system using multi-parallel data from Spanish (Es), French (Fr), Russian (Ru) and Chinese (Zh). The parallel data in these languages $X \in \{Es, Fr, Ru, Zh\}$ is used to generate word alignments between Arabic-X and X-English. These alignments are then combined to obtain multiple word alignments for Arabic-English and the final translation systems.

The motivation for this approach is two-fold. First, we believe that parallel corpora available in several languages provide a better training material for SMT systems relative to bilingual corpora. Such multi-lingual parallel corpora are becoming widely available; examples include proceedings of the United Nations in six languages (UN, 2006), European Parliament (EU, 2005; Koehn, 2003), JRC Acquis corpus (EU, 2007) and religious texts (Resnik et al., 1997). Word alignment systems

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 42–50, Prague, June 2007. ©2007 Association for Computational Linguistics

trained on different language-pairs (e.g. French-English versus Russian-English) make errors which are somewhat orthogonal. In such cases, incorrect alignment links between a sentence-pair can be corrected when a translation in a third language is available. Thus it can help resolve errors in word alignment. We combine word alignments using several bridge languages with the aim of correcting some of the alignment errors. The second advantage of this approach is that the word alignment from each bridge language can be utilized to build a phrase-based SMT system. This provides a diverse collection of translation hypotheses for MT system combination (Bangalore et al., 2002; Sim et al., 2007; Matusov et al., 2006; Macherey and Och, 2007). Finally, a side benefit of this paper is that it provides a study that compares alignment qualities and BLEU scores for models in different languages trained on parallel text which is held identical across all languages.

We show that parallel corpora in multiple languages can be exploited to improve the translation performance of a phrase-based translation system. This paper gives specific recipes for using a bridge language to construct a word alignment and for combining word alignments produced by multiple statistical alignment models.

The rest of this paper is organized as follows: Section 2 gives an overview of our framework for generating word alignments in a single language-pair. In Section 3, we describe how a bridge language may be used for producing word alignments. In Section 4, we describe a scheme to combine word alignments from several bridge languages. Section 5 describes our experimental setup and reports the alignment and translation performance. A final discussion is presented in Section 6.

2 Word Alignment Framework

A statistical translation model (Brown et al., 1993; Och and Ney, 2003) describes the relationship between a pair of sentences in the source and target languages ($\mathbf{f} = f_1^J, \mathbf{e} = e_1^I$) using a translation probability $P(\mathbf{f}|\mathbf{e})$. Alignment models introduce a hidden alignment variable $\mathbf{a} = a_1^J$ to specify a mapping between source and target words; $a_j = i$ indicates that the j^{th} source word is linked to the i^{th} target word. Alignment models assign a probability $P(\mathbf{f}, \mathbf{a}|\mathbf{e})$ to the source sentence and alignment conditioned on the target sentence. The translation probability is related to the alignment model as: $P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P_\theta(\mathbf{f}, \mathbf{a}|e)$, where θ is a set of parameters.

Given a sentence-pair $(\mathbf{f}, \mathbf{e})$, the most likely (Viterbi) word alignment is found as (Brown et al., 1993): $\hat{\mathbf{a}} = \text{argmax}_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})$. An alternate criterion is the Maximum A-Posteriori (MAP) framework (Ge, 2004; Matusov et al., 2004). We use a refinement of this technique.

Given any word alignment model, posterior probabilities can be computed as (Brown et al., 1993)

$$P(a_j = i|\mathbf{e}, \mathbf{f}) = \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{f}, \mathbf{e})\delta(i, a_j), \qquad (1)$$

where $i \in \{0, 1, ..., I\}$. The assignment $a_j = 0$ corresponds to the NULL (empty) alignment. These posterior probabilities form a matrix of size $(I+1) \times J$, where entries along each column sum to one.

The MAP alignment for each source position $j \in \{1, 2, ..., J\}$ is then computed as

$$a_{MAP}(j) = \underset{i}{\text{argmax}}\, P(a_j = i|\mathbf{e}, \mathbf{f}). \qquad (2)$$

We note that these posterior probabilities can be computed efficiently for some alignment models such as the HMM (Vogel et al., 1996; Och and Ney, 2003), Models 1 and 2 (Brown et al., 1993).

In the next two sections, we describe how posterior probabilities can be used to a) construct alignment systems from a bridge language, and b) merge several alignment systems.

3 Constructing Word Alignment Using a Bridge Language

We assume here that we have triples of sentences that are translations of each other in languages F, E, and the bridge language G: $\mathbf{f} = f_1^J, \mathbf{e} = e_1^I, \mathbf{g} = g_1^K$. Our goal is to obtain posterior probability estimates for the sentence-pair in FE: $(\mathbf{f}, \mathbf{e})$ using the posterior probability estimates for the sentence pairs in FG: $(\mathbf{f}, \mathbf{g})$ and GE: $(\mathbf{g}, \mathbf{e})$. The word alignments between the above sentence-pairs are referred to as $\mathbf{a}^{FE}$, $\mathbf{a}^{FG}$, and $\mathbf{a}^{GE}$ respectively; the notation $\mathbf{a}^{FE}$ indicates that the alignment maps a position in F to a position in E.

We first express the posterior probability as a sum over all possible translations g in G and hidden alignments a^{FG}.

$$\begin{aligned}
& P(a_j^{FE} = i|\mathbf{e}, \mathbf{f}) \\
= & \sum_{\mathbf{g}} P(a_j^{FE} = i, \mathbf{g}|\mathbf{e}, \mathbf{f}) \\
= & \sum_{\mathbf{g},k} P(a_j^{FE} = i, \mathbf{g}, a_j^{FG} = k|\mathbf{e}, \mathbf{f}) \\
= & \sum_{\mathbf{g},k} \Big\{ P(\mathbf{g}|\mathbf{e}, \mathbf{f}) P(a_j^{FG} = k|\mathbf{g}, \mathbf{e}, \mathbf{f}) \\
& \times P(a_j^{FE} = i|a_j^{FG} = k, \mathbf{g}, \mathbf{e}, \mathbf{f}) \Big\} \qquad (3)
\end{aligned}$$

We now make some assumptions to simplify the above expression. First, there is exactly one translation g in bridge language G corresponding to the sentence-pair $\mathbf{f}, \mathbf{e}$. Since $a^{GE}_{a^{FG}_j} = i = a^{FE}_j$, we can express
$P(a_j^{FE} = i|a_j^{FG} = k, \mathbf{g}, \mathbf{f}, \mathbf{e}) = P(a_k^{GE} = i|\mathbf{g}, \mathbf{e})$.
Finally, alignments in FG do not depend on E.

Under these assumptions, we arrive at the final expression for the posterior probability FE in terms of posterior probabilities for GF and EG

$$\begin{aligned}
& P(a_j^{FE} = i|\mathbf{e}, \mathbf{f}) = \qquad (4) \\
& \sum_{k=0}^{K} P(a_j^{FG} = k|\mathbf{g}, \mathbf{f}) P(a_k^{GE} = i|\mathbf{g}, \mathbf{e})
\end{aligned}$$

The above expression states that the posterior probability matrix for FE can be obtained using a *simple matrix multiplication* of posterior probability matrices for GE and FG. In this multiplication, we prepend a column to the GE matrix corresponding to $k = 0$. This probability $P(a_k^{GE} = i)$ when $k = 0$ is not assigned by the alignment model; we set it as follows

$$P(a_k^{GE} = i|k = 0) = \begin{cases} \epsilon & i = 0 \\ \frac{1-\epsilon}{I} & i \in \{1, 2, ..., I\} \end{cases}$$

The parameter ϵ controls the number of empty alignments; a higher value favors more empty alignments and vice versa. In our experiments, we set $\epsilon = 0.5$.

4 Word Alignment Combination Using Posterior Probabilities

We next show how Word Alignment Posterior Probabilities can be used for combining multiple word alignment systems. In our context, we use this procedure to combine word alignments produced using multiple bridge languages.

Suppose we have translations in bridge languages $G_1, G_2, ..., G_N$, we can generate a posterior probability matrix for FE using each of the bridge languages. In addition, we can always generate a posterior probability matrix for FE with the FE alignment model directly without using any bridge language. These $N + 1$ posterior matrices can be combined as follows. Here, the variable B indicates the bridge language. $B \in \{G_0, G_1, ..., G_N\}$; G_0 indicates the case when no bridge language is used.

$$\begin{aligned}
& P(a_j^{FE} = i|\mathbf{e}, \mathbf{f}) \qquad (5) \\
= & \sum_{l=0}^{N} P(B = G_l, a_j^{FE} = i|\mathbf{e}, \mathbf{f}) \\
= & \sum_{l=0}^{N} P(B = G_l) P(a_j^{FE} = i|G_l, \mathbf{e}, \mathbf{f}),
\end{aligned}$$

where $P(a_j^{FE} = i|G_l, j, \mathbf{e}, \mathbf{f})$ is the posterior probability when bridge language $B = G_l$. The probabilities $P(B = G_l)$ sum to one over $l \in \{0, 1, 2, ..., N\}$ and represent the prior probability of bridge language l. In our experiments, we use a uniform prior $P(B = G_l) = \frac{1}{N+1}$. Equation 5 provides us a way to combine word alignment posterior probabilites from multiple bridge languages. In our alignment framework (Section 2), we first interpolate the posterior probability matrices (Equation 5) and then extract the MAP word alignment (Equation 2) from the resulting matrix.

5 Experiments

We now present experiments to demonstrate the advantages of using bridge languages. Our experiments are performed in the open data track of the NIST Arabic-to-English (A-E) machine translation task [1].

5.1 Training and Test Data

Our approach to word alignment (Section 3) requires aligned sentences in multiple languages. For training alignment models, we use the ODS United Na-

[1] http://www.nist.gov/speech/tests/mt/

Set	# of Ar words (K)	# of sentences
dev1	48.6	2007
dev2	11.4	498
test	37.8	1610
blind	36.5	1797

Table 1: Statistics for the test data.

tions parallel data (UN, 2006) which contains parliamentary documents from 1993 onwards in all six official languages of the UN: Arabic (Ar), Chinese (Zh), English (En), French (Fr), Russian (Ru), and Spanish (Es).

We merge the NIST 2001-2005 Arabic-English evaluation sets into a pool and randomly sample this collection to create two development sets (dev1,dev2) and a test set (test) with 2007, 498, and 1610 sentences respectively. Our blind test (blind) set is the NIST part of the NIST 06 evaluation set consisting of 1797 sentences. The GALE portion of the 06 evaluation set is not used in this paper. We report results on the test and blind sets. Some statistics computed on the test data are shown in Table 1.

5.2 Alignment Model Training

For training Arabic-English alignment models, we use Chinese, French, Russian and Spanish as bridge languages. We train a model for Ar-En and 4 models each for Ar-X and X-En, where X is the bridge language. To obtain aligned sentences in these language pairs, we train 9 sentence aligners. We then train alignment models for all 9 language-pairs using a recipe consisting of 6 Model-1 iterations and 6 HMM iterations. Finally, Word Alignment Posterior Probabilities are generated over the bitext. In Table 2, we report the perplexities of the alignment models for the translation directions where either Arabic or English is predicted. There are 55M Arabic tokens and 58M English tokens. We observe that the alignment model using Spanish achieves the lowest perplexity; this value is even lower than the perplexity of the direct Arabic-English model. Perplexity is related to the hardness of the word alignment; the results suggest that bridge languages such as Spanish make alignment task easier while others do not. We stress that perplexity is not related to the alignment or the translation performance.

Bridge	Perplexity	
Lang	→ Ar	→En
None	113.8	26.1
Es	99.0	22.9
Fr	138.6	30.2
Ru	128.3	27.5
Zh	126.1	34.6

Table 2: Perplexities of the alignment models.

5.3 Bridge Language Word Alignments

Each of the 4 bridge languages is utilized for constructing a word alignment for Arabic-English. Using each bridge language X, we obtain Arabic-English word alignments in both translation directions (AE and EA). The posterior matrix for AE is obtained using AX and XE matrices while the EA matrix is obtained from EX and XA matrices (Equation 4). The AE (EA) matrices from the bridge languages are then interpolated with the AE (EA) matrix obtained from the alignment model trained directly on Arabic-English (Section 4). The MAP word alignment for AE (EA) direction is computed from the AE (EA) matrix. We next outline how these word alignments are utilized in building a phrase-based SMT system.

5.4 Phrase-based SMT system

Our phrase-based SMT system is similar to the alignment template system described in Och and Ney (2004). We first extract an inventory of phrase-pairs up to length 7 from the union of AE and EA word alignments. Various feature functions (Och and Ney, 2004) are then computed over the entries in the phrase table. 5-gram word language models in English are trained on a variety of monolingual corpora (Brants et al., 2007). Minimum Error Rate Training (MERT) (Och, 2003) under BLEU criterion is used to estimate 20 feature function weights over the larger development set (dev1).

Translation is performed using a standard dynamic programming beam-search decoder (Och and Ney, 2004). Decoding is done in two passes. An initial list of 1000-best hypotheses is generated by the decoder. This list is then rescored using Minimum Bayes-Risk (MBR) decoding (Kumar and Byrne, 2004). The MBR scaling parameter is tuned on the smaller development set (dev2).

Bridge Language	Metrics(%)					
	AE			EA		
	Prec	Rec	AER	Prec	Rec	AER
None	74.1	73.9	26.0	67.3	57.7	37.9
Es	61.7	56.3	41.1	50.0	40.2	55.4
Fr	52.9	48.0	49.7	42.3	33.6	62.5
Ru	57.4	50.8	46.1	40.2	31.6	64.6
Zh	44.3	39.3	58.3	39.7	29.9	65.9
AC1	70.0	65.0	32.6	56.8	46.4	48.9

Table 3: Alignment Performance with Bridge Languages

5.5 Alignment Results

We first report alignment performance (Table 3) of the alignment models obtained using the bridge languages. Alignment results are reported in terms of Precision (Prec), Recall (Rec) and Alignment Error Rate (AER). We report these numbers on a 94-sentence test set with translations in all six languages and human word alignments in Arabic-English. Our human word alignments do not distinguish between *Sure* and *Probable* links (Och and Ney, 2003).

In these experiments, we first identify the common subset of sentences which have translations in all six languages. Each of the 9 alignment models is then trained on this subset. We report Alignment performance in both translation directions: Arabic-to-English (AE) and English-to-Arabic (EA). The first row (None) gives the results when no bridge language is used.

Among the bridge languages, Spanish gives the best alignment for Arabic-English while Chinese results in the worst. This might be related to how different the bridge language is relative to either English or Arabic. The last row (AC1) shows the performance of the alignment obtained by combining None/Es/Fr/Ru/Zh alignments. This alignment outperforms all bridge alignments but is weaker than the alignment without any bridge language. Our hypothesis is that a good choice of interpolation weights (Equation 5) would reduce AER of the AC1 combination. However, we did not investigate these choices in this paper. We report alignment error rates here to give the readers an idea of the vastly different alignment performance using each of the bridge languages.

5.6 Translation Results

We now report translation performance of our techniques. We measure performance using the NIST implementation of case sensitive BLEU-4 on true-cased translations. We observed in experiments not reported here that results are almost identical with/without Minimum Error Rate Training ; we therefore report the results without the training. We note that the blind set is the NIST subset of the 2006 NIST evaluation set. The systems reported here are for the Unlimited Data Track in Arabic-to-English and obtain competitive performance relative to the results reported on the NIST official results page [2]

We present three sets of experiments. In Table 4, we describe the first set where all 9 alignment models are trained on nearly the same set of sentences (1.9M sentences, 57.5M words in English). This makes the alignment models in all bridge languages comparable. In the first row marked None, we do not use a bridge language. Instead, an Ar-En alignment model is trained directly on the set of sentence pairs. The next four rows give the performance of alignment models trained using the bridge languages Es, Fr, Ru and Zh respectively. For each language, we use the procedure (Section 3) to obtain the posterior probability matrix for Arabic-English from Arabic-X and X-English matrices. The row AC1 refers to alignment combination using interpolation of posterior probabilities described in Section 4. We combine posterior probability matrices from the systems in the first four rows: None, Es, Ru and Zh. We exclude the Zh system from the AC1 combination because it is found to degrade the translation performance by 0.2 points on the test set.

In the final six rows of Table 4, we show the performance of a consensus decoding technique that produces a single output hypothesis by combining translation hypotheses from multiple systems; this is an MBR-like candidate selection procedure based on BLEU correlation matrices and is described in Macherey and Och (2007). We first report performance of the consensus output by combining None systems with/without MERT. Each of the following rows provides the results from consensus decoding for adding an extra system both with/without MERT. Thus, the final row (TC1) combines transla-

[2] http://www.nist.gov/speech/tests/mt/mt06eval_official_results.html

tions from 12 systems: None, Es, Fr, Ru, Zh, AC1 with/without MERT. All entries marked with an asterisk are better than the None baseline with 95% statistical significance computed using paired bootstrap resampling (Koehn, 2004).

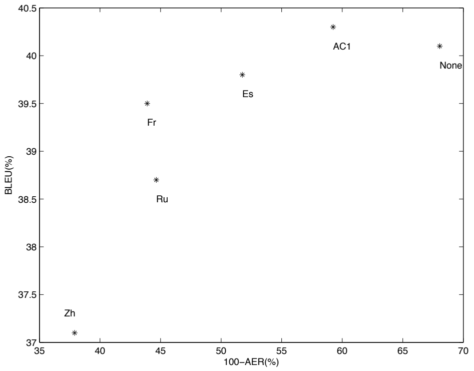


Figure 1: 100-AER (%) vs. BLEU(%) on the blind set for 6 systems from Table 3.

Figure 1 shows the plot between 100-AER% (average of EA/AE directions) and BLEU for the six systems in Table 3. We observe that AER is loosely correlated to BLEU ($\rho = 0.81$) though the relation is weak, as observed earlier by Fraser and Marcu (2006a). Among the bridge languages, Spanish gives the lowest AER/highest BLEU while Chinese results in highest AER/lowest BLEU. We can conclude that Spanish is closest to Arabic/English while Chinese is the farthest. All the bridge languages yield lower BLEU/higher AER relative to the No-Bridge baseline. Therefore, our estimate of the posterior probability (Equation 4) is always worse than the posterior probability obtained using a direct model. The alignment combination (AC1) behaves differently from other bridge systems in that it gives a higher AER and a higher BLEU relative to None baseline. We hypothesize that AC1 is different from the bridge language systems since it arises from a different process: interpolation with the direct model (None).

Both system combination techniques give improvements relative to None baseline: alignment combination AC1 gives a small gain (0.2 points) while the consensus translation TC1 results in a larger improvement (0.8 points). The last 4 rows of the table show that the performance of the hypothesis consensus steadily increases as systems get added to the None baseline. This shows that while bridge language systems are weaker than the direct model, they can provide complementary sources of evidence. To further validate this hypothesis, we compute inter-system BLEU scores between None/es and all the systems in Table 5. We observe that the baseline (None) is very dissimilar from the rest of the systems. We hypothesize that the baseline system has an alignment derived from a real alignment model while the rest of the bridge systems are derived using matrix multiplication. The low inter-system BLEU scores show that the bridge systems provide diverse hypotheses relative to the baseline and therefore contribute to gains in consensus decoding.

Bridge Lang	# Msents	BLEU (%)	
		test	blind
None	1.9	<u>52.1</u>	<u>40.1</u>
Es	1.9	51.7	39.8
Fr	1.9	51.2	39.5
Ru	1.9	50.4	38.7
Zh	1.9	48.4	37.1
AC1	1.9	52.1	40.3
Hypothesis Consensus			
None	1.9	51.9	39.8
+Es	1.9	52.2	40.0
+Fr	1.9	52.4*	40.5*
+Ru	1.9	52.8*	40.7*
+Zh	1.9	52.6*	40.6*
+AC1 = TC1	1.9	**53.0***	**40.9***

Table 4: Translation Experiments for Set 1; Results are reported on the test and blind set: (NIST portion of 2006 NIST eval set).

Ref	None	es	fr	ru	zh	AC1
None	100.0	60.0	59.8	59.7	59.5	58.7
es	59.6	100.0	79.9	69.3	67.4	70.5

Table 5: Inter-system BLEU scores (%) between None/es and all systems in Table 3.

To gain some insight about how the bridge systems help in Table 4, we present an example in Table 6. The example shows the consensus Translations and the 12 input translations for the consensus decoding. The example suggests that the inputs to the consensus decoding exhibit diversity.

Table 7 reports the second and third sets of experiments. For both sets, we first train each bridge language system X using all aligned sentences avail-

System	MERT	Hypothesis
None	N	The President of the National Conference Visit Iraqi Kurdistan Iraqi
None	Y	President of the Iraqi National Conference of Iraqi Kurdistan Visit
Es	N	President of the Iraqi National Congress to Visit Iraqi Kurdistan
Es	Y	President of the Iraqi National Congress to Visit Iraqi Kurdistan
Fr	N	President of the Iraqi National Conference Visits Iraqi Kurdistan
Fr	Y	Chairman of the Iraqi National Conference Visits Iraqi Kurdistan
Ru	N	The Chairman of the Iraqi National Conference Visits Iraqi Kurdistan
Ru	Y	Chairman of the Iraqi National Conference Visit the Iraqi Kurdistan
Zh	N	The Chairman of the Iraqi National Conference Visits Iraqi Kurdistan
Zh	Y	The Chairman of the Iraqi National Conference Visit Iraqi Kurdistan
AC1	N	President of the Iraqi National Congress to Visit Iraqi Kurdistan
AC1	Y	Chairman of the Iraqi National Congress to Visit Iraqi Kurdistan
TC1	-	The Chairman of the Iraqi National Conference Visits Iraqi Kurdistan
Ref	-	Head of Iraqi National Congress Visits Iraqi Kurdistan

Table 6: An example showing the Consensus Translation (TC1) and the 12 inputs for consensus decoding. The final row shows the reference translation.

able in Ar, En and X. In Set 2, the first row (Union) is an alignment model trained on all sentence-pairs in Ar-En which are available in at least one bridge language X. AC2 refers to alignment combination using bridge languages Es/Fr/Ru and Union. TC2 refers to the translation combination from 12 systems: Es/Fr/Ru/Zh/Union/AC2 with/without Minimum Error Rate training. Finally, the goal in Set 3 (last 3 rows) is to improve the best Arabic-English system that can be built using all available sentence pairs from the UN corpus. The first row (Direct) gives the performance of this Ar-En system; AC3 refers to alignment combination using Es/Fr/Ru and Direct. TC3 merges translations from Es/Fr/Ru/Zh/Direct/AC3. All entries marked with an asterisk (plus) are better than the Union (Direct) baseline with 95% statistical significance computed using paired bootstrap resampling (Koehn, 2004).

The motivation behind Sets 2 and 3 is to train all bridge language systems on as much bitext as possible. As a consequence, these systems give better results than the corresponding systems in Table 4. The Union system outperforms None by 1.7/1.4 BLEU points and provides a better baseline. We show under this scenario that system combination techniques AC2 and TC2 can still give smaller improvements (0.3/0.5 and 1.0/0.7 points) relative to this baseline.

As mentioned earlier, our approach requires sentence-aligned corpora. In our experiments, we use a single sentence aligner for each language pair (total of 9 aligners). Since these aligners make independent decisions on sentence boundaries, we end up with a smaller pool of sentences (1.9M) that is common across all language pairs. In contrast, a sentence aligner that makes simultaneous decisions in multiple languages would result in a larger set of common sentence pairs (close to 7M sentence pairs). Simard (1999) describes a sentence aligner of this type that improves alignment on a trilingual parallel text. Since we do not currently have access to such an aligner, we simulate that situation with Sets 2 and 3: AC2/AC3 do not insist that a sentence-pair be present in all input word alignments. We note that Set 2 is a data scenario that falls between Sets 1 and 3.

Set 3 provides the best baseline for Arabic-English based on the UN data by training on all parallel sentence-pairs. In this situation, system combination with bridge languages (AC3/TC3) gives reasonable improvements in BLEU on the test set (0.4/1.0 points) but only modest improvements (0.1/0.4 points) on the blind set. However, this does show that the bridge systems continue to provide orthogonal evidence at different operating points.

6 Discussion

We have described a simple approach to improve word alignments using bridge languages. This includes two components: a matrix multiplication to assemble a posterior probability matrix for the desired language-pair FE using a pair of posterior probability matrices FG and GE relative to a bridge language G. The second component is a recipe for combining word alignment systems by linearly in-

Bridge Lang	# Msents	BLEU (%)	
		test	blind
Es	4.7	53.7	40.9
Fr	4.7	53.2	40.7
Ru	4.5	52.4	39.9
Zh	3.4	49.7	37.9
Set 2			
Union	7.2	53.8	41.5
AC2	7.2	54.1	42.0*
TC2	-	**54.8***	**42.2***
Set 3			
Direct	7.0	53.9	42.2
AC3	9.0	54.3$^+$	42.3
TC3	-	**54.9**$^+$	**42.6**$^+$

Table 7: Translation performance for Sets 2 and 3 on test and blind:NIST portion of 2006 NIST eval set.

terpolating posterior probability matrices from different sources. In our case, these sources are multiple bridge languages. However, this method is more generally applicable for combining posterior matrices from different alignment models such as HMM and Model-4. Such an approach contrasts with the log-linear HMM/Model-4 combination proposed by Och and Ney (2003).

There has been recent work by Ayan and Dorr (2006) on combining word alignments from different alignment systems; this paper describes a maximum entropy framework for this combination. Their approach operates at the level of the alignment links and uses maximum entropy to decide whether or not to include an alignment link in the final output. In contrast, we use posterior probabilities as the interface between different alignment models. Another difference is that this maxent framework requires human word aligned data for training feature weights. We do not require any human word aligned data to train our combiner.

Another advantage of our approach is that it is based on word alignment posterior probability matrices that can be generated by any underlying alignment model. Therefore, this method can be used to combine word alignments generated by fairly dissimilar word alignment systems as long as the systems can produce posterior probabilities.

Bridge languages have been used by NLP researchers as a means to induce translation lexicons between distant languages without the need for parallel corpora (Schafer and Yarowsky, 2002; Mann and Yarowsky, 2001). Our current approach differs from these efforts in that we use bridge languages to improve word alignment quality between sentence pairs. Furthermore, we do not use linguistic insight to identify bridge languages. In our framework, a good bridge language is one that provides the best translation performance using the posterior matrix multiplication. Our experiments show that Spanish is a better bridge language relative to Chinese for Arabic-to-English translation. We speculate that if our approach was carried out on a data set with hundreds of languages, we might be able to automatically identify language families.

A downside of our approach is the requirement for exact sentence-aligned parallel data. Except for a few corpora such as UN, European Parliament etc, such a resource is hard to find. One solution is to create such parallel data by automatic translation and then retaining reliable translations by using confidence metrics (Ueffing and Ney, 2005).

Our approach to using bridge languages is extremely simple. Despite its simplicity, the system combination gives improvements in alignment and translation performance. In future work, we will consider several extensions to this framework that lead to more powerful system combination strategies using multiple bridge languages. We recall that the present approach trains bridge systems (e.g. Arabic-to-French, French-to-English) until the alignment stage and then uses these for constructing Arabic-to-English word alignment. An alternate scenario would be to build phrase-based SMT systems for Arabic-to-Spanish and Spanish-to-English, and then obtain Arabic-to-English translation by first translating from Arabic into Spanish and then Spanish into English. Such end-to-end bridge systems may lead to an even more diverse pool of hypotheses that could further improve system combination.

References

N. Ayan and B. Dorr. 2006. A maximum entropy approach to combining word alignments. In *HLT-NAACL*, New York, New York.

S. Bangalore, V. Murdock, and G. Riccardi. 2002. Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In *COLING*, Taipei, Taiwan.

L. Borin. 2000. You'll take the high road and I'll take the low road: Using a third language to improve bilingual word alignment. In *COLING*, pages 97–103, Saarbrucken, Germany.

T. Brants, A. Popat, P. Xu, F. Och, and J. Dean. 2007. Large language models in machine translation. In *EMNLP*, Prague, Czech Republic.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Y. Deng and W. Byrne. 2005. HMM word and phrase alignment for statistical machine translation. In *EMNLP*, Vancouver, Canada.

EU, 2005. *European Parliament Proceedings*. http://www.europarl.europa.eu.

EU, 2007. *JRC Acquis Corpus*. http://langtech.jrc.it/JRC-Acquis.html.

K. Filali and J. Bilmes. 2005. Leveraging multiple languages to improve statistical mt word alignments. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, San Juan, Puerto Rico.

A. Fraser and D. Marcu. 2006a. Measuring word alignment quality for statistical machine translation. Technical Report ISI-TR-616, ISI/University of Southern California.

A. Fraser and D. Marcu. 2006b. Semi-supervised training for statistical word alignment. In *ACL*, pages 769–776, Sydney, Australia.

N. Ge. 2004. Improvements in word alignments. In *Presentation given at DARPA/TIDES workshop*.

A. Ittycheriah and S. Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *EMNLP*, Vancouver, Canada.

P. Koehn, 2003. *European Parliament Proceedings, Sentence Aligned*. http://people.csail.mit.edu/koehn/publications/europarl/.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, Barcelona, Spain.

S. Kumar and W. Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *HLT-NAACL*, pages 169–176, Boston, MA, USA.

W. Macherey and F. Och. 2007. An empirical study on computing consensus translations from multiple machine translation systems. In *EMNLP*, Prague, Czech Republic.

G. Mann and D. Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *NAACL*, Pittsburgh, PA, USA.

J. Martin, R. Mihalcea, and T. Pedersen. 2005. Word alignment for languages with scarce resources. In *ACL Workshop on Building and Using Parallel Texts*, pages 65–74, Ann Arbor, MI, USA.

E. Matusov, R. Zens, and H. Ney. 2004. Symmetric word alignments for statistical machine translation. In *COLING*, Geneva, Switzerland.

E. Matusov, N. Ueffing, and H. Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *EACL*, Trento, Italy.

R. C. Moore. 2005. A discriminative framework for bilingual word alignment. In *EMNLP*, Vancouver, Canada.

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19 – 51.

F. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417 – 449.

F. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, Sapporo, Japan.

P. Resnik, M. Olsen, and M. Diab. 1997. Creating a parallel corpus from the book of 2000 tongues. In *Text Encoding Initiative 10th Anniversary User Conference*, Providence, RI, USA.

C. Schafer and D. Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*, Taipei, Taiwan.

K. C. Sim, W. J. Byrne, M. J. F. Gales, H. Sahbi, and P. C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Honolulu, HI, USA.

M. Simard. 1999. Text translation alignment: Three languages are better than two. In *EMNLP-VLC*, College Park, MD, USA.

N. Ueffing and H. Ney. 2005. Word-level confidence estimation for machine translation using phrase-based translation models. In *EMNLP*, pages 763 – 770, Vancouver, Canada.

UN, 2006. *ODS UN Parallel Corpus*. http://ods.un.org/.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM based word alignment in statistical translation. In *COLING*, pages 836–841, Copenhagen, Denmark.

Getting the structure right for word alignment: LEAF

Alexander Fraser
ISI / University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
fraser@isi.edu

Daniel Marcu
ISI / University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
marcu@isi.edu

Abstract

Word alignment is the problem of annotating parallel text with translational correspondence. Previous generative word alignment models have made structural assumptions such as the 1-to-1, 1-to-N, or phrase-based consecutive word assumptions, while previous discriminative models have either made such an assumption directly or used features derived from a generative model making one of these assumptions. We present a new generative alignment model which avoids these structural limitations, and show that it is effective when trained using both unsupervised and semi-supervised training methods.

1 Introduction

Several generative models and a large number of discriminatively trained models have been proposed in the literature to solve the problem of automatic word alignment of bitexts. The generative proposals have required unrealistic assumptions about the structure of the word alignments. Two assumptions are particularly common. The first is the 1-to-N assumption, meaning that each source word generates zero or more target words, which requires heuristic techniques in order to obtain alignments suitable for training a SMT system. The second is the consecutive word-based "phrasal SMT" assumption. This does not allow gaps, which can be used to particular advantage by SMT models which model hierarchical structure. Previous discriminative models have either made such assumptions directly or used features from a generative model making such an assumption. Our objective is to automatically produce alignments which can be used to build high quality machine translation systems. These are presumably close to the alignments that trained bilingual speakers produce. Human annotated alignments often contain M-to-N alignments, where several source words are aligned to several target words and the resulting unit can not be further decomposed. Source or target words in a single unit are sometimes nonconsecutive.

In this paper, we describe a new generative model which directly models M-to-N non-consecutive word alignments. The rest of the paper is organized as follows. The generative story is presented, followed by the mathematical formulation. Details of the unsupervised training procedure are described. The generative model is then decomposed into feature functions used in a log-linear model which is trained using a semi-supervised algorithm. Experiments show improvements in word alignment accuracy and usage of the generated alignments in hierarchical and phrasal SMT systems results in an increased BLEU score. Previous work is discussed and this is followed by the conclusion.

2 LEAF: a generative word alignment model

2.1 Generative story

We introduce a new generative story which enables the capture of non-consecutive M-to-N alignment structure. We have attempted to use the same labels as the generative story for Model 4 (Brown et

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 51–60, Prague, June 2007. ©2007 Association for Computational Linguistics

al., 1993), which we are extending.

Our generative story describes the stochastic generation of a target string f (sometimes referred to as the French string, or foreign string) from a source string e (sometimes referred to as the English string), consisting of l words. The variable m is the length of f. We generally use the index i to refer to source words (e_i is the English word at position i), and j to refer to target words.

Our generative story makes the distinction between different types of source words. There are head words, non-head words, and deleted words. Similarly, for target words, there are head words, non-head words, and spurious words. A head word is linked to zero or more non-head words; each non-head word is linked to from exactly one head word. The purpose of head words is to try to provide a robust representation of the semantic features necessary to determine translational correspondence. This is similar to the use of syntactic head words in statistical parsers to provide a robust representation of the syntactic features of a parse sub-tree.

A minimal translational correspondence consists of a linkage between a source head word and a target head word (and by implication, the non-head words linked to them). Deleted source words are not involved in a minimal translational correspondence, as they were "deleted" by the translation process. Spurious target words are also not involved in a minimal translational correspondence, as they spontaneously appeared during the generation of other target words.

Figure 1 shows a simple example of the stochastic generation of a French sentence from an English sentence, annotated with the step number in the generative story.

1. Choose the source word type.

   for each $i = 1, 2, ..., l$ choose a word type $\chi_i = -1$ (non-head word), $\chi_i = 0$ (deleted word) or $\chi_i = 1$ (head word) according to the distribution $g(\chi_i|e_i)$

   let $\chi_0 = 1$

2. Choose the identity of the head word for each non-head word.

   for each $i = 1, 2, ..., l$ if $\chi_i = -1$ choose a "linked from head word" value μ_i (the position of the head word which e_i is linked to) according to the distribution $w_{-1}(\mu_i - i|\text{class}_e(e_i))$

   for each $i = 1, 2, ..., l$ if $\chi_i = 1$ let $\mu_i = i$

   for each $i = 1, 2, ..., l$ if $\chi_i = 0$ let $\mu_i = 0$

   for each $i = 1, 2, ..., l$ if $\chi_{\mu_i} \neq 1$ return "failure"

3. Choose the identity of the generated target head word for each source head word.

   for each $i = 1, 2, ..., l$ if $\chi_i = 1$ choose τ_{i1} according to the distribution $t_1(\tau_{i1}|e_i)$

4. Choose the number of words in a target cept conditioned on the identity of the source head word and the source cept size (γ_i is 1 if the cept size is 1, and 2 if the cept size is greater).

   for each $i = 1, 2, ..., l$ if $\chi_i = 1$ choose a Foreign cept size ψ_i according to the distribution $s(\psi_i|e_i, \gamma_i)$

   for each $i = 1, 2, ..., l$ if $\chi_i < 1$ let $\psi_i = 0$

5. Choose the number of spurious words.

   choose ψ_0 according to the distribution $s_0(\psi_0|\sum_i \psi_i)$

   let $m = \psi_0 + \sum_{i=1}^{l} \psi_i$

6. Choose the identity of the spurious words.

   for each $k = 1, 2, ..., \psi_0$ choose τ_{0k} according to the distribution $t_0(\tau_{0k})$

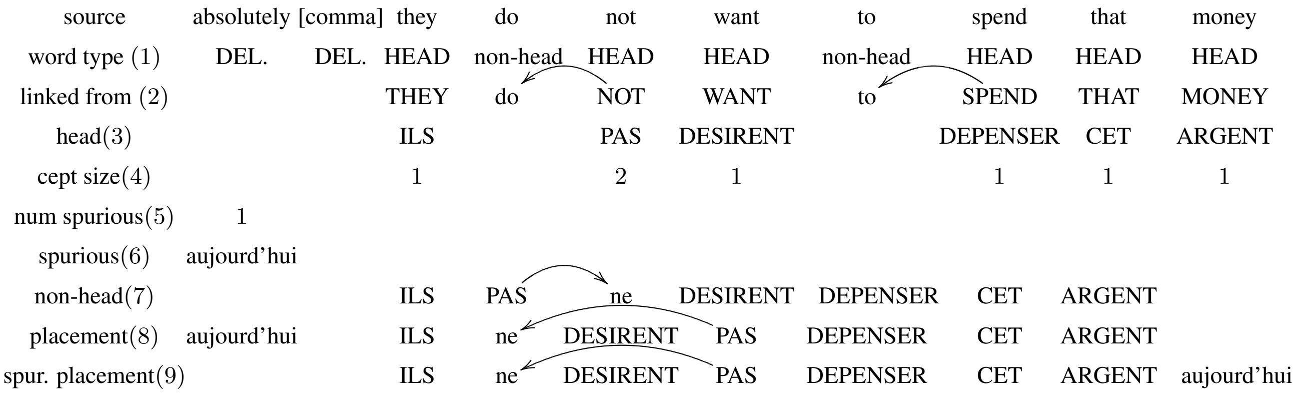
7. Choose the identity of the target non-head words linked to each target head word.

   for each $i = 1, 2, ..., l$ and for each $k = 2, 3, ..., \psi_i$ choose τ_{ik} according to the distribution $t_{>1}(\tau_{ik}|e_i, \text{class}_h(\tau_{i1}))$

8. Choose the position of the target head and non-head words.

   for each $i = 1, 2, ..., l$ and for each $k = 1, 2, ..., \psi_i$ choose a position π_{ik} as follows:

   - if $k = 1$ choose π_{i1} according to the distribution $d_1(\pi_{i1} - c_{\rho_i}|\text{class}_e(e_{\rho_i}), \text{class}_f(\tau_{i1}))$
   - if $k = 2$ choose π_{i2} according to the distribution $d_2(\pi_{i2} - \pi_{i1}|\text{class}_f(\tau_{i1}))$



source	absolutely	[comma]	they	do	not	want	to	spend	that	money
word type (1)	DEL.	DEL.	HEAD	non-head	HEAD	HEAD	non-head	HEAD	HEAD	HEAD
linked from (2)			THEY	do	NOT	WANT	to	SPEND	THAT	MONEY
head(3)			ILS		PAS	DESIRENT		DEPENSER	CET	ARGENT
cept size(4)			1		2	1		1	1	1
num spurious(5)	1									
spurious(6)	aujourd'hui									
non-head(7)			ILS	PAS	ne	DESIRENT	DEPENSER	CET	ARGENT	
placement(8)	aujourd'hui		ILS	ne	DESIRENT	PAS	DEPENSER	CET	ARGENT	
spur. placement(9)			ILS	ne	DESIRENT	PAS	DEPENSER	CET	ARGENT	aujourd'hui

Figure 1: Generative story example, (number) indicates step number

- if $k > 2$ choose π_{ik} according to the distribution $d_{>2}(\pi_{ik} - \pi_{ik-1}|\text{class}_f(\tau_{i1}))$

if any position was chosen twice, return "failure"

9. Choose the position of the spuriously generated words.

   for each $k = 1, 2, ..., \psi_0$ choose a position π_{0k} from $\psi_0 - k + 1$ remaining vacant positions in $1, 2, ..., m$ according to the uniform distribution

   let f be the string $f\pi_{ik} = \tau_{ik}$

We note that the steps which return "failure" are required because the model is deficient. Deficiency means that a portion of the probability mass in the model is allocated towards generative stories which would result in infeasible alignment structures. Our model has deficiency in the non-spurious target word placement, just as Model 4 does. It has additional deficiency in the source word linking decisions. (Och and Ney, 2003) presented results suggesting that the additional parameters required to ensure that a model is not deficient result in inferior performance, but we plan to study whether this is the case for our generative model in future work.

Given e, f and a candidate alignment a, which represents both the links between source and target head-words and the head-word connections of the non-head words, we would like to calculate $p(f, a|e)$. The formula for this is:

$$
\begin{aligned}
p(f, a|e) =& [\prod_{i=1}^{l} g(\chi_i|e_i)] \\
& [\prod_{i=1}^{l} \delta(\chi_i, -1) w_{-1}(\mu_i - i|\text{class}_e(e_i))] \\
& [\prod_{i=1}^{l} \delta(\chi_i, 1) t_1(\tau_{i1}|e_i)] \\
& [\prod_{i=1}^{l} \delta(\chi_i, 1) s(\psi_i|e_i, \gamma_i)] \\
& [s_0(\psi_0|\sum_{i=1}^{l} \psi_i)] \\
& [\prod_{k=1}^{\psi_0} t_0(\tau_{0k})] \\
& [\prod_{i=1}^{l} \prod_{k=2}^{\psi_i} t_{>1}(\tau_{ik}|e_i, \text{class}_h(\tau_{i1}))] \\
& [\prod_{i=1}^{l} \prod_{k=1}^{\psi_i} D_{ik}(\pi_{ik})]
\end{aligned}
$$

where:

$\delta(i, i')$ is the Kronecker delta function which is equal to 1 if $i = i'$ and 0 otherwise.

ρ_i is the position of the closest English head word to the left of the word at i or 0 if there is no such word.

$\text{class}_e(e_i)$ is the word class of the English word at position i, $\text{class}_f(f_j)$ is the word class of the French word at position j, $\text{class}_h(f_j)$ is the word class of the French head word at position j.

p_0 and p_1 are parameters describing the probability of not generating and of generating a target spurious word from each non-spurious target word, $p_0 + p_1 = 1$.

$$m' = \sum_{i=1}^{l} \psi_i \tag{1}$$

$$s_0(\psi_0|m') = \binom{m'}{\psi_0} p_0^{m'-\psi_0} p_1^{\psi_0} \tag{2}$$

$$D_{ik}(j) = \begin{cases} d_1(j - c_{\rho_i}|\text{class}_e(e_{\rho_i}), \text{class}_f(\tau_{ik})) \\ \qquad \text{if } k = 1 \\ d_2(j - \pi_{i1}|\text{class}_f(\tau_{ik})) \\ \qquad \text{if } k = 2 \\ d_{>2}(j - \pi_{ik-1}|\text{class}_f(\tau_{ik})) \\ \qquad \text{if } k > 2 \end{cases} \tag{3}$$

$$\gamma_i = \min(2, \sum_{i'=1}^{l} \delta(\mu_{i'}, i)) \tag{4}$$

$$c_i = \begin{cases} \text{ceiling}(\sum_{k=1}^{\psi_i} \pi_{ik}/\psi_i) & \text{if } \psi_i \neq 0 \\ 0 & \text{if } \psi_i = 0 \end{cases} \tag{5}$$

The alignment structure used in many other models can be modeled using special cases of this framework. We can express the 1-to-N structure of models like Model 4 by disallowing $\chi_i = -1$, while for 1-to-1 structure we both disallow $\chi_i = -1$ and deterministically set $\psi_i = \chi_i$. We can also specialize our generative story to the consecutive word M-to-N alignments used in "phrase-based" models, though in this case the conditioning of the generation decisions would be quite different. This involves adding checks on source and target connection geometry to the generative story which, if violated, would return "failure"; naturally this is at the cost of additional deficiency.

2.2 Unsupervised Parameter Estimation

We can perform maximum likelihood estimation of the parameters of this model in a similar fashion to that of Model 4 (Brown et al., 1993), described thoroughly in (Och and Ney, 2003). We use Viterbi training (Brown et al., 1993) but neighborhood estimation (Al-Onaizan et al., 1999; Och and Ney, 2003) or "pegging" (Brown et al., 1993) could also be used.

To initialize the parameters of the generative model for the first iteration, we use bootstrapping from a 1-to-N and a M-to-1 alignment. We use the intersection of the 1-to-N and M-to-1 alignments to establish the head word relationship, the 1-to-N alignment to delineate the target word cepts, and the M-to-1 alignment to delineate the source word cepts.

In bootstrapping, a problem arises when we encounter infeasible alignment structure where, for instance, a source word generates target words but no link between any of the target words and the source word appears in the intersection, so it is not clear which target word is the target head word. To address this, we consider each of the N generated target words as the target head word in turn and assign this configuration 1/N of the counts.

For each iteration of training we search for the Viterbi solution for millions of sentences. Evidence that inference over the space of all possible alignments is intractable has been presented, for a similar problem, in (Knight, 1999). Unlike phrase-based SMT, left-to-right hypothesis extension using a beam decoder is unlikely to be effective because in word alignment reordering is not limited to a small local window and so the necessary beam would be very large. We are not aware of admissible or inadmissible search heuristics which have been shown to be effective when used in conjunction with a search algorithm similar to A* search for a model predicting over a structure like ours. Therefore we use a simple local search algorithm which operates on complete hypotheses.

(Brown et al., 1993) defined two local search operations for their 1-to-N alignment models 3, 4 and 5. All alignments which are reachable via these operations from the starting alignment are considered. One operation is to change the generation decision for a French word to a different English word (move), and the other is to swap the generation decision for two French words (swap). All possible operations are tried and the best is chosen. This is repeated. The search is terminated when no operation results in an improvement. (Och and Ney, 2003) discussed efficient implementation.

In our model, because the alignment structure is richer, we define the following operations: move French non-head word to new head, move English non-head word to new head, swap heads of two French non-head words, swap heads of two English non-head words, swap English head word links of two French head words, link English word to French word making new head words, unlink English and French head words. We use multiple restarts to try to reduce search errors. (Germann et al., 2004; Marcu and Wong, 2002) have some similar operations without the head word distinction.

3 Semi-supervised parameter estimation

Equation 6 defines a log-linear model. Each feature function h_m has an associated weight λ_m. Given a vector of these weights λ, the alignment search problem, i.e. the search to return the best alignment $\hat{a}$ of the sentences e and f according to the model, is specified by Equation 7.

$$p_\lambda(f,a|e) = \frac{\exp(\sum_m \lambda_m h_m(a,e,f))}{\sum_{a',f'} \exp(\sum_m \lambda_m h_m(a',e,f'))} \quad (6)$$

$$\hat{a} = \operatorname*{argmax}_a \sum_m \lambda_m h_m(f,a,e) \quad (7)$$

We decompose the new generative model presented in Section 2 in both translation directions to provide the initial feature functions for our log-linear model, features 1 to 10 and 16 to 25 in Table 1.

We use backoffs for the translation decisions (features 11 and 26 and the HMM translation tables which are features 12 and 27) and the target cept size distributions (features 13, 14, 28 and 29 in Table 1), as well as heuristics which directly control the number of unaligned words we generate (features 15 and 30 in Table 1).

We use the semi-supervised EMD algorithm (Fraser and Marcu, 2006b) to train the model. The initial M-step bootstraps parameters as described in Section 2.2 from a M-to-1 and a 1-to-N alignment. We then perform the D-step following (Fraser and Marcu, 2006b). Given the feature function parameters estimated in the M-step and the feature function weights λ determined in the D-step, the E-step searches for the Viterbi alignment for the full training corpus.

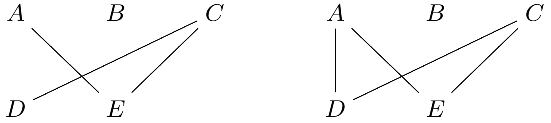


Figure 2: Two alignments with the same translational correspondence

We use $1 -$ F-Measure as our error criterion. (Fraser and Marcu, 2006a) established that it is important to tune α (the trade-off between Precision and Recall) to maximize performance. In working with LEAF, we discovered a methodological problem with our baseline systems, which is that two alignments which have the same translational correspondence can have different F-Measures. An example is shown in Figure 2.

To overcome this problem we fully interlinked the transitive closure of the undirected bigraph formed by each alignment hypothesized by our baseline alignment systems[1]. This operation maps the alignment shown to the left in Figure 2 to the alignment shown to the right. This operation does not change the collection of phrases or rules extracted from a hypothesized alignment, see, for instance, (Koehn et al., 2003). Working with this fully interlinked representation we found that the best settings of α were $\alpha = 0.1$ for the Arabic/English task and $\alpha = 0.4$ for the French/English task.

4 Experiments

4.1 Data Sets

We perform experiments on two large alignments tasks, for Arabic/English and French/English data sets. Statistics for these sets are shown in Table 2. All of the data used is available from the Linguistic Data Consortium except for the French/English

[1] All of the gold standard alignments were fully interlinked as distributed. We did not modify the gold standard alignments.

1	$chi(\chi_i \mid e_i)$ source word type	9	$d_2(\triangle j \mid \text{class}_f(f_j))$ movement for left-most target non-head word
2	$\mu(\triangle i \mid \text{class}_e(e_i))$ choosing a head word	10	$d_{>2}(\triangle j \mid \text{class}_f(f_j))$ movement for subsequent target non-head words
3	$t_1(f_j \mid e_i)$ head word translation	11	$t(f_j \mid e_i)$ translation without dependency on word-type
4	$s(\psi_i \mid e_i, \gamma_i)$ ψ_i is number of words in target cept	12	$t(f_j \mid e_i)$ translation table from final HMM iteration
5	$s_0(\psi_0 \mid \sum_i \psi_i)$ number of unaligned target words	13	$s(\psi_i \mid \gamma_i)$ target cept size without dependency on source head word e
6	$t_0(f_j)$ identity of unaligned target words	14	$s(\psi_i \mid e_i)$ target cept size without dependency on γ_i
7	$t_{>1}(f_j \mid e_i, \text{class}_h(\tau_{i1}))$ non-head word translation	15	target spurious word penalty
8	$d_1(\triangle j \mid \text{class}_e(e_\rho), \text{class}_f(f_j))$ movement for target head words	16-30	(same features, other direction)

Table 1: Feature functions

gold standard alignments which are available from the authors.

4.2 Experiments

To build all alignment systems, we start with 5 iterations of Model 1 followed by 4 iterations of HMM (Vogel et al., 1996), as implemented in GIZA++ (Och and Ney, 2003).

For all non-LEAF systems, we take the best performing of the "union", "refined" and "intersection" symmetrization heuristics (Och and Ney, 2003) to combine the 1-to-N and M-to-1 directions resulting in a M-to-N alignment. Because these systems do not output fully linked alignments, we fully link the resulting alignments as described at the end of Section 3. The reader should recall that this does not change the set of rules or phrases that can be extracted using the alignment.

We perform one main comparison, which is of semi-supervised systems, which is what we will use to produce alignments for SMT. We compare semi-supervised LEAF with a previous state of the art semi-supervised system (Fraser and Marcu, 2006b). We performed translation experiments on the alignments generated using semi-supervised training to verify that the improvements in F-Measure result in increases in BLEU.

We also compare the unsupervised LEAF system with GIZA++ Model 4 to give some idea of the performance of the unsupervised model. We made an effort to optimize the free parameters of GIZA++, while for unsupervised LEAF there are no free parameters to optimize. A single iteration of unsupervised LEAF[2] is compared with heuristic symmetrization of GIZA++'s extension of Model 4 (which was run for four iterations). LEAF was bootstrapped as described in Section 2.2 from the HMM Viterbi alignments.

Results for the experiments on the French/English data set are shown in Table 3. We ran GIZA++ for four iterations of Model 4 and used the "refined" heuristic (line 1). We ran the baseline semi-supervised system for two iterations (line 2), and in contrast with (Fraser and Marcu, 2006b) we found that the best symmetrization heuristic for this system was "union", which is most likely due to our use of fully linked alignments which was discussed at the end of Section 3. We observe that LEAF unsupervised (line 3) is competitive with GIZA++ (line 1), and is in fact competitive with the baseline semi-supervised result (line 2). We ran the LEAF semi-supervised system for two iterations (line 4). The best result is the LEAF semi-supervised system, with a gain of 1.8 F-Measure over the LEAF unsupervised system.

For French/English translation we use a state of the art phrase-based MT system similar to (Och and Ney, 2004; Koehn et al., 2003). The translation test data is described in Table 2. We use two trigram language models, one built using the English portion of the training data and the other built using additional English news data. The BLEU scores reported in this work are calculated using lowercased and tokenized data. For semi-supervised LEAF the gain of 0.46 BLEU over the semi-supervised baseline is not statistically significant (a gain of 0.78 BLEU would be required), but LEAF semi-supervised compared with GIZA++ is significant, with a gain of 1.23 BLEU. We note that this shows a large gain in trans-

[2] Unsupervised LEAF is equivalent to using the log-linear model and setting $\lambda_m = 1$ for $m = 1$ to 10 and $m = 16$ to 25, while setting $\lambda_m = 0$ for other values of m.

		Arabic/English		French/English	
		A	E	F	E
Training	Sents	6,609,162		2,842,184	
	Words	147,165,003	168,301,299	75,794,254	67,366,819
	Vocab	642,518	352,357	149,568	114,907
	Singletons	256,778	158,544	60,651	47,765
Align Discr.	Sents	1,000		110	
	Words	26,882	37,635	1,888	1,726
	Links	39,931		2,292	
Align Test	Sents	83		110	
	Words	1,510	2,030	1,899	1,716
	Links	2,131		2,176	
Trans. Dev	Sents	728 (4 references)		833 (1 reference)	
	Words	18,255	22.0K to 24.6K	20,562	17,454
Trans. Test	Sents	1,056 (4 references)		2,380 (1 reference)	
	Words	28,505	35.8K to 38.1K	58,990	49,182

Table 2: Data sets

lation quality over that obtained using GIZA++ because BLEU is calculated using only a single reference for the French/English task.

Results for the Arabic/English data set are also shown in Table 3. We used a large gold standard word alignment set available from the LDC. We ran GIZA++ for four iterations of Model 4 and used the "union" heuristic. We compare GIZA++ (line 1) with one iteration of the unsupervised LEAF model (line 2). The unsupervised LEAF system is worse than four iterations of GIZA++ Model 4. We believe that the features in LEAF are too high dimensional to use for the Arabic/English task without the backoffs available in the semi-supervised models. The baseline semi-supervised system (line 3) was run for three iterations and the resulting alignments were combined with the "union" heuristic. We ran the LEAF semi-supervised system for two iterations. The best result is the LEAF semi-supervised system (line 4), with a gain of 5.4 F-Measure over the baseline semi-supervised system.

For Arabic/English translation we train a state of the art hierarchical model similar to (Chiang, 2005) using our Viterbi alignments. The translation test data used is described in Table 2. We use two trigram language models, one built using the English portion of the training data and the other built using additional English news data. The test set is from the NIST 2005 translation task. LEAF had the best performance scoring 1.43 BLEU better than the baseline semi-supervised system, which is statistically significant.

5 Previous Work

The LEAF model is inspired by the literature on generative modeling for statistical word alignment and particularly by Model 4 (Brown et al., 1993). Much of the additional work on generative modeling of 1-to-N word alignments is based on the HMM model (Vogel et al., 1996). (Toutanova et al., 2002) and (Lopez and Resnik, 2005) presented a variety of refinements of the HMM model particularly effective for low data conditions. (Deng and Byrne, 2005) described work on extending the HMM model using a bigram formulation to generate 1-to-N alignment structure. The common thread connecting these works is their reliance on the 1-to-N approximation, while we have defined a generative model which does not require use of this approximation, at the cost of having to rely on local search.

There has also been work on generative models for other alignment structures. (Wang and Waibel, 1998) introduced a generative story based on extension of the generative story of Model 4. The alignment structure modeled was "consecutive M to non-consecutive N". (Marcu and Wong, 2002) defined the Joint model, which modeled consecutive word M-to-N alignments. (Matusov et al., 2004) presented a model capable of modeling 1-to-N and M-to-1 alignments (but not arbitrary M-to-N alignments) which was bootstrapped from Model 4. LEAF directly models non-consecutive M-to-N alignments.

One important aspect of LEAF is its symmetry. (Och and Ney, 2003) invented heuristic symmetriza-

	French/English		Arabic/English	
System	F-Measure ($\alpha = 0.4$)	BLEU	F-measure ($\alpha = 0.1$)	BLEU
GIZA++	73.5	30.63	75.8	51.55
(Fraser and Marcu, 2006b)	74.1	31.40	79.1	52.89
LEAF unsupervised	74.5		72.3	
LEAF semi-supervised	76.3	31.86	84.5	54.34

Table 3: Experimental Results

tion of the output of a 1-to-N model and a M-to-1 model resulting in a M-to-N alignment, this was extended in (Koehn et al., 2003). We have used insights from these works to help determine the structure of our generative model. (Zens et al., 2004) introduced a model featuring a symmetrized lexicon. (Liang et al., 2006) showed how to train two HMM models, a 1-to-N model and a M-to-1 model, to agree in predicting all of the links generated, resulting in a 1-to-1 alignment with occasional rare 1-to-N or M-to-1 links. We improve on these works by choosing a new structure for our generative model, the head word link structure, which is both symmetric and a robust structure for modeling of non-consecutive M-to-N alignments.

In designing LEAF, we were also inspired by dependency-based alignment models (Wu, 1997; Alshawi et al., 2000; Yamada and Knight, 2001; Cherry and Lin, 2003; Zhang and Gildea, 2004). In contrast with their approaches, we have a very flat, one-level notion of dependency, which is bilingually motivated and learned automatically from the parallel corpus. This idea of dependency has some similarity with hierarchical SMT models such as (Chiang, 2005).

The discriminative component of our work is based on a plethora of recent literature. This literature generally views the discriminative modeling problem as a supervised problem involving the combination of heuristically derived feature functions. These feature functions generally include the prediction of some type of generative model, such as the HMM model or Model 4. A discriminatively trained 1-to-N model with feature functions specifically designed for Arabic was presented in (Ittycheriah and Roukos, 2005). (Lacoste-Julien et al., 2006) created a discriminative model able to model 1-to-1, 1-to-2 and 2-to-1 alignments for which the best results were obtained using features based on symmetric HMMs trained to agree, (Liang et al., 2006), and intersected Model 4. (Ayan and Dorr, 2006) defined a discriminative model which learns how to combine the predictions of several alignment algorithms. The experiments performed included Model 4 and the HMM extensions of (Lopez and Resnik, 2005). (Moore et al., 2006) introduced a discriminative model of 1-to-N and M-to-1 alignments, and similarly to (Lacoste-Julien et al., 2006) the best results were obtained using HMMs trained to agree and intersected Model 4. LEAF is not bound by the structural restrictions present either directly in these models, or in the features derived from the generative models used. We also iterate the generative/discriminative process, which allows the discriminative predictions to influence the generative model.

Our work is most similar to work using discriminative log-linear models for alignment, which is similar to discriminative log-linear models used for the SMT decoding (translation) problem (Och and Ney, 2002; Och, 2003). (Liu et al., 2005) presented a log-linear model combining IBM Model 3 trained in both directions with heuristic features which resulted in a 1-to-1 alignment. (Fraser and Marcu, 2006b) described symmetrized training of a 1-to-N log-linear model and a M-to-1 log-linear model. These models took advantage of features derived from both training directions, similar to the symmetrized lexicons of (Zens et al., 2004), including features derived from the HMM model and Model 4. However, despite the symmetric lexicons, these models were only able to optimize the performance of the 1-to-N model and the M-to-1 model separately, and the predictions of the two models required combination with symmetrization heuristics. We have overcome the limitations of that work by defining new feature functions, based on the LEAF generative model, which score non-consecutive M-to-N alignments so that the final performance criterion can be optimized directly.

6 Conclusion

We have found a new structure over which we can robustly predict which directly models translational correspondence commensurate with how it is used in hierarchical SMT systems. Our new generative model, LEAF, is able to model alignments which consist of M-to-N non-consecutive translational correspondences. Unsupervised LEAF is comparable with a strong baseline. When coupled with a discriminative training procedure, the model leads to increases between 3 and 9 F-score points in alignment accuracy and 1.2 and 2.8 BLEU points in translation accuracy over strong French/English and Arabic/English baselines.

7 Acknowledgments

This work was partially supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022. We would like to thank the USC Center for High Performance Computing and Communications.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John D. Lafferty, I. Dan Melamed, David Purdy, Franz J. Och, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation, final report, JHU workshop.

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1):45–60.

Necip Fazil Ayan and Bonnie J. Dorr. 2006. A maximum entropy approach to combining word alignments. In *Proceedings of HLT-NAACL*, pages 96–103, New York.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of ACL*, pages 88–95, Sapporo, Japan.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, MI.

Yonggang Deng and William Byrne. 2005. Hmm word and phrase alignment for statistical machine translation. In *Proceedings of HLT-EMNLP*, Vancouver, Canada.

Alexander Fraser and Daniel Marcu. 2006a. Measuring word alignment quality for statistical machine translation. In *Technical Report ISI-TR-616*, ISI/University of Southern California.

Alexander Fraser and Daniel Marcu. 2006b. Semi-supervised training for statistical word alignment. In *Proceedings of COLING-ACL*, pages 769–776, Sydney, Australia.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2004. Fast decoding and optimal decoding for machine translation. *Artificial Intelligence*, 154(1-2):127–143.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT-EMNLP*, pages 89–96, Vancouver, Canada.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133, Edmonton, Canada.

Simon Lacoste-Julien, Dan Klein, Ben Taskar, and Michael Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of HLT-NAACL*, pages 112–119, New York, NY.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*, New York.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL*, pages 459–466, Ann Arbor, MI.

Adam Lopez and Philip Resnik. 2005. Improved hmm alignment models for languages with scarce resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 83–86, Ann Arbor, MI.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*, pages 133–139, Philadelphia, PA.

Evgeny Matusov, Richard Zens, and Hermann Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proceedings of COLING*, Geneva, Switzerland.

Robert C. Moore, Wen-Tau Yih, and Andreas Bode. 2006. Improved discriminative bilingual word alignment. In *Proceedings of COLING-ACL*, pages 513–520, Sydney, Australia.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295–302, Philadelphia, PA.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(1):417–449.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167, Sapporo, Japan.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to hmm-based statistical word alignment models. In *Proceedings of EMNLP*, Philadelphia, PA.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, pages 836–841, Copenhagen, Denmark.

Ye-Yi Wang and Alex Waibel. 1998. Modeling with structures in statistical machine translation. In *Proceedings of COLING-ACL*, volume 2, pages 1357–1363, Montreal, Canada.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL*, pages 523–530, Toulouse, France.

Richard Zens, Evgeny Matusov, and Hermann Ney. 2004. Improved word alignment using a symmetric lexicon model. In *Proceedings of COLING*, Geneva, Switzerland.

Hao Zhang and Daniel Gildea. 2004. Syntax-based alignment: Supervised or unsupervised? In *Proceedings of COLING*, Geneva, Switzerland.

Improving Statistical Machine Translation using Word Sense Disambiguation

Marine CARPUAT marine@cs.ust.hk
Dekai WU[*] dekai@cs.ust.hk

Human Language Technology Center
HKUST
Department of Computer Science and Engineering
University of Science and Technology, Clear Water Bay, Hong Kong

Abstract

We show for the first time that incorporating the predictions of a word sense disambiguation system within a typical phrase-based statistical machine translation (SMT) model consistently improves translation quality across *all* three different IWSLT Chinese-English test sets, as well as producing statistically significant improvements on the larger NIST Chinese-English MT task—and moreover *never* hurts performance on any test set, according not only to BLEU but to *all eight* most commonly used automatic evaluation metrics. Recent work has challenged the assumption that word sense disambiguation (WSD) systems are useful for SMT. Yet SMT translation quality still obviously suffers from inaccurate lexical choice. In this paper, we address this problem by investigating a new strategy for integrating WSD into an SMT system, that performs *fully phrasal multi-word* disambiguation. Instead of directly incorporating a Senseval-style WSD system, we redefine the WSD task to match the exact same phrasal translation disambiguation task faced by phrase-based SMT systems. Our results provide the first known empirical evidence that lexical semantics are indeed useful for SMT, despite claims to the contrary.

1 Introduction

Common assumptions about the role and usefulness of word sense disambiguation (WSD) models in full-scale statistical machine translation (SMT) systems have recently been challenged.

On the one hand, in previous work (Carpuat and Wu, 2005b) we obtained disappointing results when using the predictions of a Senseval WSD system in conjunction with a standard word-based SMT system: we reported slightly lower BLEU scores despite trying to incorporate WSD using a number of apparently sensible methods. These results cast doubt on the assumption that sophisticated dedicated WSD systems that were developed independently from any particular NLP application can easily be integrated into a SMT system so as to improve translation quality through stronger models of context and rich linguistic information. Rather, it has been argued, SMT systems have managed to achieve significant improvements in translation quality without directly addressing translation disambiguation as a WSD task. Instead, translation disambiguation decisions are made indirectly, typically using only word surface forms and very local contextual information, forgoing the much richer linguistic information that WSD systems typically take advantage of.

On the other hand, error analysis reveals that the performance of SMT systems still suffers from inaccurate lexical choice. In subsequent empirical studies, we have shown that SMT systems perform much worse than dedicated WSD models, both supervised

[*] This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under GALE Contract No. HR0011-06-C-0023, and by the Hong Kong Research Grants Council (RGC) research grants RGC6083/99E, RGC6256/00E, and DAG03/04.EG09. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 61–72, Prague, June 2007. ©2007 Association for Computational Linguistics

and unsupervised, on a Senseval WSD task (Carpuat and Wu, 2005a), and therefore suggest that WSD should have a role to play in state-of-the-art SMT systems. In addition to the Senseval shared tasks, which have provided standard sense inventories and data sets, WSD research has also turned increasingly to designing specific models for a particular application. For instance, Vickrey *et al.* (2005) and Specia (2006) proposed WSD systems designed for French to English, and Portuguese to English translation respectively, and present a more optimistic outlook for the use of WSD in MT, although these WSD systems have not yet been integrated nor evaluated in full-scale machine translation systems.

Taken together, these seemingly contradictory results suggest that improving SMT lexical choice accuracy remains a key challenge to improve current SMT quality, and that it is still unclear what is the most appropriate integration framework for the WSD models in SMT.

In this paper, we present first results with a new architecture that integrates a state-of-the-art WSD model into phrase-based SMT so as to perform *multi-word phrasal* lexical disambiguation, and show that this new WSD approach not only produces gains across *all* available Chinese-English IWSLT06 test sets for all eight commonly used automated MT evaluation metrics, but also produces statistically significant gains on the much larger NIST Chinese-English task. The main difference between this approach and several of our earlier approaches as described in Carpuat and Wu (2005b) and subsequently Carpuat *et al.* (2006) lies in the fact that we focus on repurposing the WSD system for multi-word phrase-based SMT. Rather than using a generic Senseval WSD model as we did in Carpuat and Wu (2005b), here both the WSD training and the WSD predictions are integrated into the phrase-based SMT framework. Furthermore, rather than using a single word based WSD approach to augment a phrase-based SMT model as we did in Carpuat *et al.* (2006) to improve BLEU and NIST scores, here the WSD training and predictions operate on full multi-word phrasal units, resulting in significantly more reliable and consistent gains as evaluted by many other translation accuracy metrics as well. Specifically:

- Instead of using a Senseval system, we redefine the WSD task to be exactly the same as lexical choice task faced by the multi-word phrasal translation disambiguation task faced by the phrase-based SMT system.

- Instead of using predefined senses drawn from manually constructed sense inventories such as HowNet (Dong, 1998), our WSD for SMT system directly disambiguates between all phrasal translation candidates seen during SMT training.

- Instead of learning from manually annotated training data, our WSD system is trained on the same corpora as the SMT system.

However, despite these adaptations to the SMT task, the core sense disambiguation task remains pure WSD:

- The rich context features are typical of WSD and almost never used in SMT.

- The dynamic integration of context-sensitive translation probabilities is not typical of SMT.

- Although it is embedded in a real SMT system, the WSD task is exactly the same as in recent and coming Senseval Multilingual Lexical Sample tasks (e.g., Chklovski *et al.* (2004)), where sense inventories represent the semantic distinctions made by another language.

We begin by presenting the WSD module and the SMT integration technique. We then show that incorporating it into a standard phrase-based SMT baseline system *consistently improves translation quality* across all three different test sets from the Chinese-English IWSLT text translation evaluation, as well as on the larger NIST Chinese-English translation task. Depending on the metric, the individual gains are sometimes modest, but remarkably, incorporating WSD *never* hurts, and helps enough to always make it a worthwile additional component in an SMT system. Finally, we analyze the reasons for the improvement.

2 Problems in context-sensitive lexical choice for SMT

To the best of our knowledge, there has been no previous attempt at integrating a state-of-the-art WSD system for fully phrasal multi-word lexical choice into phrase-based SMT, with evaluation of the resulting system on a translation task. While there are many evaluations of WSD quality, in particular the Senseval series of shared tasks (Kilgarriff and Rosenzweig (1999), Kilgarriff (2001), Mihalcea *et al.* (2004)), very little work has been done to address the actual integration of WSD in realistic SMT applications.

To fully integrate WSD into phrase-based SMT, it is necessary to perform lexical disambiguation on *multi-word phrasal* lexical units; in contrast, the model reported in Cabezas and Resnik (2005) can only perform lexical disambiguation on *single words*. Like the model proposed in this paper, Cabezas and Resnik attempted to integrate phrase-based WSD models into decoding. However, although they reported that incorporating these predictions via the Pharaoh XML markup scheme yielded a small improvement in BLEU score over a Pharaoh baseline on a single Spanish-English translation data set, we have determined empirically that applying their single-word based model to several Chinese-English datasets does *not* yield systematic improvements on most MT evaluation metrics (Carpuat and Wu, 2007). The single-word model has the disadvantage of forcing the decoder to choose between the baseline phrasal translation probabilities versus the WSD model predictions for single words. In addition, the single-word model does not generalize to WSD for phrasal lexical choice, as overlapping spans cannot be specified with the XML markup scheme. Providing WSD predictions for phrases would require committing to a phrase segmentation of the input sentence before decoding, which is likely to hurt translation quality.

It is also necessary to focus directly on translation accuracy rather than other measures such as alignment error rate, which may not actually lead to improved translation quality; in contrast, for example, Garcia-Varea *et al.* (2001) and Garcia-Varea *et al.* (2002) show improved alignment error rate with a maximum entropy based context-dependent lexical choice model, but not improved translation accuracy. In contrast, our evaluation in this paper is conducted on the actual decoding task, rather than intermediate tasks such as word alignment. Moreover, in the present work, *all* commonly available automated MT evaluation metrics are used, rather than only BLEU score, so as to maintain a more balanced perspective.

Another problem in the context-sensitive lexical choice in SMT models of Garcia Varea *et al.* is that their feature set is insufficiently rich to make much better predictions than the SMT model itself. In contrast, our WSD-based lexical choice models are designed to directly model the lexical choice in the actual translation direction, and take full advantage of not residing strictly within the Bayesian source-channel model in order to benefit from the much richer Senseval-style feature set this facilitates.

Garcia Varea *et al.* found that the best results are obtained when the training of the context-dependent translation model is fully incorporated with the EM training of the SMT system. As described below, the training of our new WSD model, though not incorporated within the EM training, is also far more closely tied to the SMT model than is the case with traditional standalone WSD models.

In contrast with Brown *et al.* (1991), our approach incorporates the predictions of state-of-the-art WSD models that use rich contextual features for any phrase in the input vocabulary. In Brown *et al.*'s early study of WSD impact on SMT performance, the authors reported improved translation quality on a French to English task, by choosing an English translation for a French word based on the single contextual feature which is reliably discriminative. However, this was a pilot study, which is limited to words with exactly two translation candidates, and it is not clear that the conclusions would generalize to more recent SMT architectures.

3 Problems in translation-oriented WSD

The close relationship between WSD and SMT has been emphasized since the emergence of WSD as an independent task. However, most of previous research has focused on using multilingual resources typically used in SMT systems to improve WSD accuracy, e.g., Dagan and Itai (1994), Li and Li (2002), Diab (2004). In contrast, this paper focuses on the converse goal of using WSD models to improve actual translation quality.

Recently, several researchers have focused on designing WSD systems for the specific purpose of translation. Vickrey *et al.* (2005) train a logistic regression WSD model on data extracted from automatically word aligned parallel corpora, but evaluate on a blank filling task, which is essentially an evaluation of WSD accuracy. Specia (2006) describes an inductive logic programming-based WSD system, which was specifically designed for the purpose of Portuguese to English translation, but this system was also only evaluated on WSD accuracy, and not integrated in a full-scale machine translation system.

Ng *et al.* (2003) show that it is possible to use automatically word aligned parallel corpora to train accurate supervised WSD models. The purpose of the study was to lower the annotation cost for supervised WSD, as suggested earlier by Resnik and Yarowsky (1999). However this result is also encouraging for the integration of WSD in SMT, since it suggests that accurate WSD can be achieved using training data of the kind needed for SMT.

4 Building WSD models for phrase-based SMT

4.1 WSD models for every phrase in the input vocabulary

Just like for the baseline phrase translation model, WSD models are defined for every phrase in the input vocabulary. Lexical choice in SMT is naturally framed as a WSD problem, so the first step of integration consists of defining a WSD model for every phrase in the SMT input vocabulary.

This differs from traditional WSD tasks, where the WSD target is a single content word. Senseval for instance has either lexical sample or all word tasks. The target words for both categories of Senseval WSD tasks are typically only content words—primarily nouns, verbs, and adjectives—while in the context of SMT, we need to translate entire sentences, and therefore have a WSD model not only for every word in the input sentences, regardless of their POS tag, but for every phrase, including tokens such as articles, prepositions and even punctuation. Further empirical studies have suggested that including WSD predictions for those longer phrases is a key factor to help the decoder produce better translations (Carpuat and Wu, 2007).

4.2 WSD uses the same sense definitions as the SMT system

Instead of using pre-defined sense inventories, the WSD models disambiguate between the SMT translation candidates. In order to closely integrate WSD predictions into the SMT system, we need to formulate WSD models so that they produce features that can directly be used in translation decisions taken by the SMT system. It is therefore necessary for the WSD and SMT systems to consider exactly the same translation candidates for a given word in the input language.

Assuming a standard phrase-based SMT system (e.g., Koehn *et al.* (2003)), WSD senses are thus either words or phrases, as learned in the SMT phrasal translation lexicon. Those "sense" candidates are very different from those typically used even in dedicated WSD tasks, even in the multilingual Senseval tasks. Each candidate is a phrase that is not necessarily a syntactic noun or verb phrase as in manually compiled dictionaries. It is quite possible that distinct "senses" in our WSD for SMT system could be considered synonyms in a traditional WSD framework, especially in monolingual WSD.

In addition to the consistency requirements for integration, this requirement is also motivated by empirical studies, which show that predefined translations derived from sense distinctions defined in monolingual ontologies do not match translation distinction made by human translators (Specia *et al.*, 2006).

4.3 WSD uses the same training data as the SMT system

WSD training does not require any other resources than SMT training, nor any manual sense annotation. We employ supervised WSD systems, since Senseval results have amply demonstrated that supervised models significantly outperform unsupervised approaches (see for instance the English lexical sample tasks results described by Mihalcea *et al.* (2004)).

Training examples are annotated using the phrase alignments learned during SMT training. Every input language phrase is sense-tagged with its aligned output language phrase in the parallel corpus. The phrase alignment method used to extract the WSD training data therefore depends on the one used by the SMT system. This presents the advantage of training WSD and SMT models on exactly the same data, thus eliminating domain mismatches between Senseval data and parallel corpora. But most importantly, this allows WSD training data to be generated entirely automatically, since the parallel corpus is automatically phrase-aligned in order to learn the SMT phrase bilexicon.

4.4 The WSD system

The word sense disambiguation subsystem is modeled after the best performing WSD system in the Chinese lexical sample task at Senseval-3 (Carpuat *et al.*, 2004).

The features employed are typical of WSD and are therefore far richer than those used in most SMT systems. The feature set consists of position-sensitive, syntactic, and local collocational features, since these features yielded the best results when combined in a naïve Bayes model on several Senseval-2 lexical sample tasks (Yarowsky and Florian, 2002). These features scale easily to the bigger vocabulary and sense candidates to be considered in a SMT task.

The Senseval system consists of an ensemble of four combined WSD models:

The first model is a naïve Bayes model, since Yarowsky and Florian (2002) found this model to be the most accurate classifier in a comparative study on a subset of Senseval-2 English lexical sample data.

The second model is a maximum entropy model (Jaynes, 1978), since Klein and Manning (Klein and Manning, 2002) found that this model yielded higher accuracy than naïve Bayes in a subsequent comparison of WSD performance.

The third model is a boosting model (Freund and Schapire, 1997), since boosting has consistently turned in very competitive scores on related tasks such as named entity classification. We also use the Adaboost.MH algorithm.

The fourth model is a Kernel PCA-based model (Wu *et al.*, 2004). Kernel Principal Component Analysis or KPCA is a nonlinear kernel method for extracting nonlinear principal components from vector sets where, conceptually, the n-dimensional input vectors are nonlinearly mapped from their original space R^n to a high-dimensional feature space F where linear PCA is performed, yielding a transform by which the input vectors can be mapped nonlinearly to a new set of vectors (Schölkopf *et al.*, 1998). WSD can be performed by a Nearest Neighbor Classifier in the high-dimensional KPCA feature space.

All these classifiers have the ability to handle large numbers of sparse features, many of which may be irrelevant. Moreover, the maximum entropy and boosting models are known to be well suited to handling features that are highly interdependent.

4.5 Integrating WSD predictions in phrase-based SMT architectures

It is non-trivial to incorporate WSD into an existing phrase-based architecture such as Pharaoh (Koehn, 2004), since the decoder is not set up to easily accept multiple translation probabilities that are dynamically computed in context-sensitive fashion.

For every *phrase* in a given SMT input sentence, the WSD probabilities can be used as additional feature in a loglinear translation model, in combination with typical context-independent SMT bilexicon probabilities.

We overcome this obstacle by devising a calling architecture that reinitializes the decoder with dynamically generated lexicons on a per-sentence basis.

Unlike a n-best reranking approach, which is limited by the lexical choices made by the decoder using only the baseline context-independent translation probabilities, our method allows the system to make full use of WSD information for all competing phrases at all decoding stages.

5 Experimental setup

The evaluation is conducted on two standard Chinese to English translation tasks. We follow standard machine translation evaluation procedure using automatic evaluation metrics. Since our goal is to evaluate translation quality, we use standard MT evaluation methodology and do not evaluate the accuracy of the WSD model independently.

Table 1: Evaluation results on the IWSLT06 dataset: integrating the WSD translation predictions improves BLEU, NIST, METEOR, WER, PER, CDER and TER across all 3 different available test sets.

Test Set	**Exper.**	**BLEU**	**NIST**	**METEOR**	**METEOR (no syn)**	**TER**	**WER**	**PER**	**CDER**
Test 1	SMT	42.21	7.888	65.40	63.24	40.45	45.58	37.80	40.09
	SMT+WSD	**42.38**	**7.902**	**65.73**	**63.64**	**39.98**	**45.30**	**37.60**	**39.91**
Test 2	SMT	41.49	8.167	66.25	63.85	40.95	46.42	37.52	40.35
	SMT+WSD	**41.97**	**8.244**	**66.35**	**63.86**	**40.63**	**46.14**	**37.25**	**40.10**
Test 3	SMT	49.91	9.016	73.36	70.70	35.60	40.60	32.30	35.46
	SMT+WSD	**51.05**	**9.142**	**74.13**	**71.44**	**34.68**	**39.75**	**31.71**	**34.58**

Table 2: Evaluation results on the NIST test set: integrating the WSD translation predictions improves BLEU, NIST, METEOR, WER, PER, CDER and TER

Exper.	**BLEU**	**NIST**	**METEOR**	**METEOR (no syn)**	**TER**	**WER**	**PER**	**CDER**
SMT	20.41	7.155	60.21	56.15	76.76	88.26	61.71	70.32
SMT+WSD	**20.92**	**7.468**	**60.30**	**56.79**	**71.34**	**83.87**	**57.29**	**67.38**

5.1 Data set

Preliminary experiments are conducted using training and evaluation data drawn from the multilingual BTEC corpus, which contains sentences used in conversations in the travel domain, and their translations in several languages. A subset of this data was made available for the IWSLT06 evaluation campaign (Paul, 2006); the training set consists of 40000 sentence pairs, and each test set contains around 500 sentences. We used only the pure text data, and not the speech transcriptions, so that speech-specific issues would not interfere with our primary goal of understanding the effect of integrating WSD in a full-scale phrase-based model.

A larger scale evaluation is conducted on the standard NIST Chinese-English test set (MT-04), which contains 1788 sentences drawn from newswire corpora, and therefore of a much wider domain than the IWSLT data set. The training set consists of about 1 million sentence pairs in the news domain.

Basic preprocessing was applied to the corpus. The English side was simply tokenized and case-normalized. The Chinese side was word segmented using the LDC segmenter.

5.2 Baseline SMT system

Since our focus is not on a specific SMT architecture, we use the off-the-shelf phrase-based decoder Pharaoh (Koehn, 2004) trained on the IWSLT training set. Pharaoh implements a beam search decoder for phrase-based statistical models, and presents the advantages of being freely available and widely used.

The phrase bilexicon is derived from the intersection of bidirectional IBM Model 4 alignments, obtained with GIZA++ (Och and Ney, 2003), augmented to improve recall using the grow-diag-final heuristic. The language model is trained on the English side of the corpus using the SRI language modeling toolkit (Stolcke, 2002).

The loglinear model weights are learned using Chiang's implementation of the maximum BLEU training algorithm (Och, 2003), both for the baseline, and the WSD-augmented system. Due to time constraints, this optimization was only conducted on the IWSLT task. The weights used in the WSD-augmented NIST model are based on the best IWSLT model. Given that the two tasks are quite different, we expect further improvements on the WSD-augmented system after running maximum BLEU optimization for the NIST task.

6 Results and discussion

Using WSD predictions in SMT yields better translation quality on *all* test sets, as measured by *all eight* commonly used automatic evaluation metrics.

Table 3: Translation examples with and without WSD for SMT, drawn from IWSLT data sets.

Input	请 转 乘 中央 线 。
Ref.	Please transfer to the Chuo train line.
SMT	Please turn to the Central Line.
SMT+WSD	Please transfer to Central Line.
Input	车票 在 车上 买 吗 ？
Ref.	Do I pay on the bus?
SMT	Please get on the bus?
SMT+WSD	I buy a ticket on the bus?
Input	需要 预订 吗 ？
Ref.	Do I need a reservation?
SMT	I need a reservation?
SMT+WSD	Do I need a reservation?
Input	我 想 再 确认 一下 这 张 票 的 预订 。
Ref.	I want to reconfirm this ticket.
SMT	I would like to reconfirm a flight for this ticket.
SMT+WSD	I would like to reconfirm my reservation for this ticket.
Input	步行 可以 到 那里 吗 ？
Ref.	Can I get there on foot?
SMT	Is there on foot?
SMT+WSD	Can I get there on foot?
Input	我 有 另外 一个 约会 ， 所以 请 快点 。
Ref.	I have another appointment, so please hurry.
SMT	I have an appointment for a, so please hurry.
SMT+WSD	I have another appointment, so please hurry.
Input	对不起 。 你 能 告诉 我 到 百老汇 的 路 吗 ？
Ref.	Excuse me. Could you tell me the way to Broadway?
SMT	Could you tell me the way to Broadway? I am sorry.
SMT+WSD	Excuse me, could you tell me the way to Broadway?
Input	对不起 ， 我 想 开 一个 账 户 。
Ref.	Excuse me, I want to open an account.
SMT	Excuse me, I would like to have an account.
SMT+WSD	Excuse me, I would like to open an account.

The results are shown in Table 1 for IWSLT and Table 2 for the NIST task. Paired bootstrap resampling shows that the improvements on the NIST test set are statistically significant at the 95% level.

Remarkably, integrating WSD predictions helps all the very different metrics. In addition to the widely used BLEU (Papineni *et al.*, 2002) and NIST (Doddington, 2002) scores, we also evaluate translation quality with the recently proposed Meteor (Banerjee and Lavie, 2005) and four edit-distance style metrics, Word Error Rate (WER), Position-independent word Error Rate (PER) (Tillmann *et al.*, 1997), CDER, which allows block reordering (Leusch *et al.*, 2006), and Translation Edit Rate (TER) (Snover *et al.*, 2006). Note that we report Meteor scores computed both with and without using WordNet synonyms to match translation candidates and references, showing that the improvement is not due to context-independent synonym matches at evaluation time.

Comparison of the 1-Best decoder output with and without the WSD feature shows that the sentences differ by one or more token respectively for 25.49%, 30.40% and 29.25% of IWSLT test sets 1,

Table 4: Translation examples with and without WSD for SMT, drawn from the NIST test set.

Input	没有 任何 议员 投票 反对 他 。
SMT	Without any congressmen voted against him.
SMT+WSD	No congressmen voted against him.
Input	俄 在 车臣 实行 的 政策 以及 对 独 联 体 邻国 的 态度 更 是 令 美国 担忧 。
SMT	Russia’s policy in Chechnya and CIS neighbors attitude is even more worried that the United States.
SMT+WSD	Russia’s policy in Chechnya and its attitude toward its CIS neighbors cause the United States still more anxiety.
Input	至于 美国 的 人权 状况 呢 ？
SMT	As for the U.S. human rights conditions?
SMT+WSD	As for the human rights situation in the U.S.?
Input	我 参 拜 是 为了 祈求 日本 的 和平 与 繁荣 。
SMT	The purpose of my visit to Japan is pray for peace and prosperity.
SMT+WSD	The purpose of my visit is to pray for peace and prosperity for Japan.
Input	为 防范 恐怖活动， 洛杉矶 警方 采取 了 前所未有 的 严密 保安 措施 。
SMT	In order to prevent terrorist activities Los Angeles, the police have taken unprecedented tight security measures.
SMT+WSD	In order to prevent terrorist activities Los Angeles, the police to an unprecedented tight security measures.

2 and 3, and 95.74% of the NIST test set.

Tables 3 and 4 show examples of translations drawn from the IWSLT and NIST test sets respectively.

A more detailed analysis reveals WSD predictions give better rankings and are more discriminative than baseline translation probabilities, which helps the final translation in three different ways.

- The rich context features help rank the correct translation first with WSD while it is ranked lower according to baseline translation probability scores .

- Even when WSD and baseline translation probabilities agree on the top translation candidate, the stronger WSD scores help override wrong language model predictions.

- The strong WSD scores for phrases help the decoder pick longer phrase translations, while using baseline translation probabilities often translate those phrases in smaller chunks that include a frequent (and incorrect) translation candidate.

For instance, the top 4 Chinese sentences in Table 4, are better translated by the WSD-augmented system because the WSD scores help the decoder to choose longer phrases. In the first example, the phrase “没有 任何” is correctly translated as a whole as “No” by the WSD-augmented system, while the baseline translates each word separately yielding an incorrect translation. In the following three examples, the WSD system encourages the decoder to translate the long phrases “更 是 令 美国 担忧”, “美国 的 人权 状况”, and “祈求 日本 的 和平 与 繁荣” as single units, while the baseline introduces errors by breaking them down into shorter phrases.

The last sentence in the table shows an example where the WSD predictions do not help the baseline system. The translation quality is actually much worse, since the verb “采取” is incorrectly translated as “to”, despite the fact that the top candidate predicted by the WSD system alone is the much better translation “has taken”, but with a relatively low probability of 0.509.

7 Conclusion

We have shown for the first time that integrating multi-word phrasal WSD models into phrase-based

SMT consistently helps on all commonly available automated translation quality evaluation metrics on all three different test sets from the Chinese-English IWSLT06 text translation task, and yields statistically significant gains on the larger NIST Chinese-English task. It is important to note that the WSD models *never* hurt translation quality, and always yield individual gains of a level that makes their integration always worthwile.

We have proposed to consistently integrate WSD models both during training, where sense definitions and sense-annotated data are automatically extracted from the word-aligned parallel corpora from SMT training, and during testing, where the phrasal WSD probabilities are used by the SMT system just like all the other lexical choice features.

Context features are derived from state-of-the-art WSD models, and the evaluation is conducted on the actual translation task, rather than intermediate tasks such as word alignment.

It is to be emphasized that this approach does not merely consist of adding a source sentence feature in the log linear model for translation. On the contrary, it remains a real WSD task, defined just as in the Senseval Multilingual Lexical Sample tasks (e.g., Chklovski *et al.* (2004)). Our model makes use of typical WSD features that are almost never used in SMT systems, and requires a dynamically created translation lexicon on a per-sentence basis.

To our knowledge this constitues the first attempt at fully integrating state-of-the-art WSD with conventional phrase-based SMT. Unlike previous approaches, the WSD targets are not only single words, but *multi-word phrases*, just as in the SMT system. This means that WSD senses are unusually predicted not only for a limited set of single words or very short phrases, but for all phrases of arbitrarily length that are in the SMT translation lexicon. The single word approach, as we reported in Carpuat *et al.* (2006), improved BLEU and NIST scores for phrase-based SMT, but subsequent detailed empirical studies we have performed since then suggest that single word WSD approaches are less successful when evaluated under all other MT metrics (Carpuat and Wu, 2007). Thus, fully phrasal WSD predictions for longer phrases, as reported in this paper, are particularly important to improve translation quality.

The results reported in this paper cast new light on the WSD vs. SMT debate, suggesting that a close integration of WSD and SMT decisions should be incorporated in a SMT model that successfully uses WSD predictions. Our objective here is to demonstrate that this technique works for the widest possible class of models, so we have chosen as the baseline the most widely used phrase-based SMT model. Our positive results suggest that our experiments could be tried on other current statistical MT models, especially the growing family of tree-structured SMT models employing stochastic transduction grammars of various sorts (Wu and Chiang, 2007). For instance, incorporating WSD predictions into an MT decoder based on inversion transduction grammars (Wu, 1997)—such as the Bracketing ITG based models of Wu (1996), Zens *et al.* (2004), or Cherry and Lin (2007)—would present an intriguing comparison with the present work. It would also be interesting to assess whether a more grammatically structured statistical MT model that is less reliant on an n-gram language model, such as the syntactic ITG based "grammatical channel" translation model of (Wu and Wong, 1998), could make more effective use of WSD predictions.

References

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgement. In *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43th Annual Meeting of the Association of Computational Linguistics (ACL-2005)*, Ann Arbor, Michigan, June 2005.

Peter Brown, Stephen Della Pietra, Vincent Della Pietra, and Robert Mercer. Word-sense disambiguation using statistical methods. In *Proceedings of 29th meeting of the Association for Computational Linguistics*, pages 264–270, Berkeley, California, 1991.

Clara Cabezas and Philip Resnik. Using WSD techniques for lexical selection in statistical machine translation. Technical report, Institute for Advanced Computer Studies, University of Maryland, 2005.

Marine Carpuat and Dekai Wu. Evaluating the word sense disambiguation performance of statistical machine translation. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP)*, pages 122–127, Jeju Island, Republic of Korea, 2005.

Marine Carpuat and Dekai Wu. Word sense disambiguation vs. statistical machine translation. In *Proceedings of the annual meeting of the association for computational linguistics (ACL-05)*, Ann Arbor, Michigan, 2005.

Marine Carpuat and Dekai Wu. How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. Forthcoming, 2007.

Marine Carpuat, Weifeng Su, and Dekai Wu. Augmenting ensemble classification for word sense disambiguation with a Kernel PCA model. In *Proceedings of Senseval-3, Third International Workshop on Evaluating Word Sense Disambiguation Systems*, Barcelona, July 2004. SIGLEX, Association for Computational Linguistics.

Marine Carpuat, Yihai Shen, Xiaofeng Yu, and Dekai Wu. Toward integrating word sense and entity disambiguation into statistical machine translation. In *Third International Workshop on Spoken Language Translation (IWSLT 2006)*, Kyoto, November 2006.

Colin Cherry and Dekang Lin. Inversion Transduction Grammar for joint phrasal translation modeling. In Dekai Wu and David Chiang, editors, *NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 17–24, Rochester, NY, April 2007.

Timothy Chklovski, Rada Mihalcea, Ted Pedersen, and Amruta Purandare. The Senseval-3 multilingual English-Hindi lexical sample task. In *Proceedings of Senseval-3, Third International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 5–8, Barcelona, Spain, July 2004. SIGLEX, Association for Computational Linguistics.

Ido Dagan and Alon Itai. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics*, 20(4):563–596, 1994.

Mona Diab. Relieving the data acquisition bottleneck in word sense disambiguation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Human Language Technology conference (HLT-2002)*, San Diego, CA, 2002.

Zhendong Dong. Knowledge description: what, how and who? In *Proceedings of International Symposium on Electronic Dictionary*, Tokyo, Japan, 1998.

Yoram Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Journal of Computer and System Sciences, 55(1)*, pages 119–139, 1997.

Ismael Garcia-Varea, Franz Och, Hermann Ney, and Francisco Casacuberta. Refined lexicon models for statistical machine translation using a maximum entropy approach. In *Proceedings of the 39th annual meeting of the association for computational linguistics (ACL-01)*, Toulouse, France, 2001.

Ismael Garcia-Varea, Franz Och, Hermann Ney, and Francisco Casacuberta. Efficient integration of maximum entropy lexicon models within the training of statistical alignment models. In *Proceedings of AMTA-2002*, pages 54–63, Tiburon, California, October 2002.

E.T. Jaynes. *Where do we Stand on Maximum Entropy?* MIT Press, Cambridge MA, 1978.

Adam Kilgarriff and Joseph Rosenzweig. Framework and results for English Senseval. *Computers and the Humanities*, 34(1):15–48, 1999. Special issue on SENSEVAL.

Adam Kilgarriff. English lexical sample task description. In *Proceedings of Senseval-2, Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 17–20, Toulouse, France, July 2001. SIGLEX, Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. Conditional structure versus conditional estimation in NLP models. In *Proceedings of EMNLP-2002, Conference on Empirical Methods in Natural Language*

Processing, pages 9–16, Philadelphia, July 2002. SIGDAT, Association for Computational Linguistics.

Philipp Koehn, Franz Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of HLT/NAACL-2003*, Edmonton, Canada, May 2003.

Philipp Koehn. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *6th Conference of the Association for Machine Translation in the Americas (AMTA)*, Washington, DC, September 2004.

Gregor Leusch, Nicola Ueffing, and Hermann Ney. Efficient MT evaluation using block movements. In *Proceedings of EACL-2006 (11th Conference of the European Chapter of the Association for Computational Linguistics)*, pages 241–248, Trento, Italy, April 2006.

Cong Li and Hang Li. Word translation disambiguation using bilingual bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 343–351, 2002.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgariff. The Senseval-3 English lexical sample task. In *Proceedings of Senseval-3, Third International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 25–28, Barcelona, Spain, July 2004. SIGLEX, Association for Computational Linguistics.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of ACL-03, Sapporo, Japan*, pages 455–462, 2003.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52, 2003.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

Michael Paul. Overview of the IWSLT06 evaluation campaign. In *Third International Workshop on Spoken Language Translation (IWSLT 2006)*, Kyoto, November 2006.

Philip Resnik and David Yarowsky. Distinguising systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural Language Engineering*, 5(2):113–133, 1999.

Bernhard Schölkopf, Alexander Smola, and Klaus-Rober Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5), 1998.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231, Boston, MA, 2006. Association for Machine Translation in the Americas.

Lucia Specia, Maria das Graĉas Volpe Nunes, Gabriela Castelo Branco Ribeiro, and Mark Stevenson. Multilingual versus monolingual WSD. In *EACL-2006 Workshop on Making Sense of Sense: Bringing Psycholinguistics and Computational Linguistics Together*, pages 33–40, Trento, Italy, April 2006.

Lucia Specia. A hybrid relational approach for WSD—first results. In *Proceedings of the COLING/ACL 06 Student Research Workshop*, pages 55–60, Sydney, July 2006. ACL.

Andreas Stolcke. SRILM—an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, Denver, Colorado, September 2002.

Christoph Tillmann, Stefan Vogel, Hermann Ney, A. Zubiaga, and H. Sawaf. Accelerated DP-based search for statistical translation. In *Proceedings of Eurospeech'97*, pages 2667–2670, Rhodes, Greece, 1997.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. Word-sense disambiguation for machine translation. In *Joint Human Language Technology conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, 2005.

Dekai Wu and David Chiang, editors. *NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation (SSST)*. Association for Computational Linguistics, Rochester, NY, USA, April 2007.

Dekai Wu and Hongsing Wong. Machine translation with a stochastic grammatical channel. In *Proceedings of COLING-ACL'98*, Montreal,Canada, August 1998.

Dekai Wu, Weifeng Su, and Marine Carpuat. A Kernel PCA method for superior word sense disambiguation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, July 2004.

Dekai Wu. A polynomial-time algorithm for statistical machine translation. In *Proceedings of 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, California, June 1996.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, 1997.

David Yarowsky and Radu Florian. Evaluating sense disambiguation across diverse parameter spaces. *Natural Language Engineering*, 8(4):293–310, 2002.

Richard Zens, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. Reordering constraints for phrase-based statistical machine translation. In *20th International Conference on Computational Linguistics (COLING-2004)*, Geneva, August 2004.

Large Margin Synchronous Generation and its Application to Sentence Compression

Trevor Cohn and **Mirella Lapata**
School of Informatics
University of Edinburgh
Edinburgh, United Kingdom
{tcohn,mlap}@inf.ed.ac.uk

Abstract

This paper presents a tree-to-tree transduction method for text rewriting. Our model is based on synchronous tree substitution grammar, a formalism that allows local distortion of the tree topology and can thus naturally capture structural mismatches. We describe an algorithm for decoding in this framework and show how the model can be trained discriminatively within a large margin framework. Experimental results on sentence compression bring significant improvements over a state-of-the-art model.

1 Introduction

Recent years have witnessed increasing interest in text-to-text generation methods for many natural language processing applications ranging from text summarisation to question answering and machine translation. At the heart of these methods lies the ability to perform rewriting operations according to a set of prespecified constraints. For example, text simplification identifies which phrases or sentences in a document will pose reading difficulty for a given user and substitutes them with simpler alternatives (Carroll et al., 1999). Sentence compression produces a summary of a single sentence that retains the most important information while remaining grammatical (Jing, 2000).

Ideally, we would like a text-to-text rewriting system that is not application specific. Given a parallel corpus of training examples, we should be able to learn rewrite rules and how to combine them in order to generate new text. A great deal of previous work has focused on the rule induction problem (Barzilay and McKeown, 2001; Pang et al., 2003; Lin and Pantel, 2001; Shinyama et al., 2002), whereas relatively little emphasis has been placed on the actual generation task (Quirk et al., 2004). A notable exception is sentence compression for which end-to-end rewriting systems are commonly developed (Knight and Marcu, 2002; Turner and Charniak, 2005; Galley and McKeown, 2007; Riezler et al., 2003; McDonald, 2006). The appeal of this task lies in its simplified formulation as a single rewrite operation, namely word deletion (Knight and Marcu, 2002).

Solutions to the compression task have been cast mostly in a supervised learning setting (but see Clarke and Lapata (2006a), Hori and Furui (2004), and Turner and Charniak (2005) for unsupervised methods). Rewrite rules are learnt from a parsed parallel corpus and subsequently used to find the best compression from the set of all possible compressions for a given sentence. A common assumption is that the tree structures representing long sentences and their compressions are *isomorphic*. Consequently, the models are not generally applicable to other text rewriting problems since they cannot readily handle structural mismatches and more complex rewriting operations such as substitutions or insertions. A related issue is that the tree structure of the compressed sentences is often poor; most algorithms delete words or constituents without paying too much attention to the structure of the compressed sentence. However, without an explicit generation mechanism that allows tree transformations, there is no guarantee that the compressions will have well-formed syntactic structures. And it will not be easy to process them for subsequent generation or analysis tasks.

In this paper we present a text-to-text rewriting

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 73–82, Prague, June 2007. ©2007 Association for Computational Linguistics

model that scales to *non-isomorphic* cases and can thus naturally account for structural and lexical divergences. Our approach is inspired by *synchronous tree substitution grammar* (STSG, Eisner (2003)) a formalism that allows local distortion of the tree topology. We show how such a grammar can be induced from a parallel corpus and propose a large margin model for the rewriting task which can be viewed as a weighted tree-to-tree transducer. Our learning framework makes use of the algorithm put forward by Tsochantaridis et al. (2005) which efficiently learns a prediction function to minimise a given loss function. Experiments on sentence compression show significant improvements over the state-of-the-art. Beyond sentence compression and related text-to-text generation problems (*e.g.,* paraphrasing), our model is generally applicable to tasks involving structural mapping. Examples include machine translation (Eisner, 2003) or semantic parsing (Zettlemoyer and Collins, 2005).

2 Related Work

Knight and Marcu (2002) proposed a noisy-channel formulation of sentence compression based on synchronous context-free grammar (SCFG). The latter is a generalisation of the context-free grammar (CFG) formalism to simultaneously produce strings in two languages. In the case of sentence compression, the grammar rules have two right hand sides, one corresponding to the *source* (long) sentence and the other to its *target* compression. The synchronous derivations are learnt from a parallel corpus and their probabilities are estimated generatively.

Given a long sentence, l, the aim is to find the corresponding compressed sentence, s, which maximises $P(s)P(l|s)$ (here $P(s)$ is the source model and $P(l|s)$ the channel model.) Modifications of this model are reported in Turner and Charniak (2005) and Galley and McKeown (2007) with improved results. The channel model is limited to tree deletion and does not allow any type of tree re-organisation.

Non-isomorphic tree structures are common when translating between languages. It is therefore not surprising that most previous work on tree rewriting falls within the realm of machine translation. Proposals include Eisner's (2003) synchronous tree substitution grammar (STSG), Melamed's (2004) multitext grammar, and Graehl and Knight's (2004) tree-to-tree transducers. Despite differences in formalism, all these approaches model the translation process using tree-based probabilistic transduction rules. The grammar induction process requires EM training which can be computationally expensive especially if all synchronous rules are considered.

Our work formulates sentence compression in the framework of STSG (Eisner, 2003). We propose a novel grammar induction algorithm that does not require EM training and is coupled with a separate large margin training process (Tsochantaridis et al., 2005) for weighting each rule. McDonald (2006) also presents a sentence compression model that uses a discriminative large margin algorithm. However, we differ in two important respects. First, our generation algorithm is more powerful, performing complex tree transformations, whereas McDonald only considers simple word deletion. Being tree-based, the generation algorithm is better able to preserve the grammaticality of the compressed output. Second, our model can be tuned to a wider range of loss functions (*e.g.,*tree-based measures).

3 Problem Formulation

We formulate sentence compression as an instance of the general problem of learning a mapping from input patterns $\mathbf{x} \in \mathcal{X}$ to discrete structured objects $\mathbf{y} \in \mathcal{Y}$. Our training sample consists of a parallel corpus of input (uncompressed) and output (compressed) pairs $(x_1, y_1) \ldots (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ and our task is to predict a target labelled tree $\mathbf{y}$ from a source labelled tree $\mathbf{x}$. As we describe below, $\mathbf{y}$ is not precisely a target tree, but instead derivations which generate both the source and the target tree. We model the dependency between $\mathbf{x}$ and $\mathbf{y}$ as a weighted STSG. Grammar rules are of the form $\langle X, Y \rangle \rightarrow \langle \gamma, \alpha, \beta \rangle$ where γ and α are *elementary trees* composed of a mixture of terminal and non-terminals rooted with non-terminals X and Y respectively, and β is a set of variable correspondences between pairs of frontier non-terminals in γ and α. A grammar rule specifies that we can substitute the trees γ and α for corresponding X and Y nodes in the source and target trees respectively. For example, the rule:

$$\langle NP,\ NP \rangle \rightarrow \langle [DT_{\boxed{1}}\ ADJP\ NN_{\boxed{2}}]_{NP}, [DT_{\boxed{1}}\ NN_{\boxed{2}}]_{NP} \rangle$$

allows adjective phrases to be dropped from the source tree within an NP. The indices $_{\boxed{x}}$ are used to specify the variable correspondences, β.

Each grammar rule has a score from which the overall score of a compression $\mathbf{y}$ for sentence $\mathbf{x}$ can be derived. These scores are learnt discriminatively using the large margin technique proposed by Tsochantaridis et al. (2005). The synchronous rules are combined using a chart-based parsing algorithm (Eisner, 2003) to generate the derivation (i.e., compressed tree) with the highest score.

We begin by describing our STSG generation algorithm in Section 3.1. We next explain how a synchronous grammar is induced from a parallel corpus of original sentences and their compressions (Section 3.2) and give the details of our learning framework (Section 3.3).

3.1 Generation

Generation aims to find the best target tree for a given source tree using the transformations specified by the synchronous grammar. (We discuss how we obtain this grammar in the following section.)

$$\mathbf{y}^* = \max_{\mathbf{y} \in \mathcal{Y}} score(\mathbf{x}, \mathbf{y}; \mathbf{w}) \qquad (1)$$

where $\mathbf{y}$ ranges over all target derivations (and therefore trees), $\mathbf{w}$ is a parameter vector and $score(\cdot)$ is an objective function measuring the quality of the derivation. In common with many parsing methods, we encounter a problem with *spurious ambiguity*: i.e., there may be many *derivations* (sequences of rule applications) which produce the same target tree. Ideally we would sum up the scores over all these derivations, however for the sake of tractability we instead take the maximum score. This allows us to pose the maximisation problem over *derivations* rather than target trees.

The generation algorithm uses a dynamic program defined over the constituents in the source tree as shown in Figure 1 (see also Eisner (2003)). The algorithm makes the assumption that the scoring function decomposes with the derivation, such that a partial score can be evaluated at each step, i.e., $score(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{r \in \mathbf{y}} score(r; \mathbf{w})$ where r are the rules used in the derivation. This method builds a *chart* of the best scoring partial derivation for each source subtree headed by a given target non-terminal. The inductive step is applied recursively bottom-up, and involves applying a grammar rule to a node in the source tree. Rules with substitution variables in their frontier are scored with reference to the chart for the matching nodes and target non-terminal categories. Once the process is complete, we can read the best score from the chart cell for the root node, and the best derivation can be constructed by traversing *back-pointers* also stored in the chart. This is illustrated in Figure 2 where the rule $\langle VP, VP \rangle \rightarrow \langle [[\text{is}_{AUX}\, ADJP_{\boxed{1}}]_{VP}\, CC\, VP]_{VP}, [\text{is}_{AUX}\, NP_{\boxed{1}}]_{VP} \rangle$ is applied to the top VP node. The score of the resulting tree would reference the chart to calculate the score for the best target tree at the $ADJP$ node with syntactic category NP.

```
for all nodes, $n$, in source tree (bottom-up) do
    for all rules, $r$ with left side matching node, $n_r = n$ do
        $s = score(r)$
        for all variables $v$ in $r$ do
            $score = score + chart[n_v, c_v]$
        end for
        update $chart[n, c_r]$ with score, $s$, if better than current
    end for
end for
$c_{best} = \text{argmax}_c\, chart[root, c]$
find best derivation using back-pointers from $(root, c_{best})$
```

Figure 1: Generation algorithm to find the best derivation. n_r and n_v are the source nodes indexed by the rule's source side (root and variable), while c_r and c_v are the non-terminal categories of the rule's target side (root and variable).

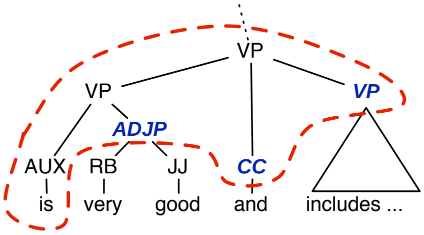


Figure 2: Example of a rule application during generation. The dashed area shows a matching rule for the VP node.

3.2 Grammar Induction

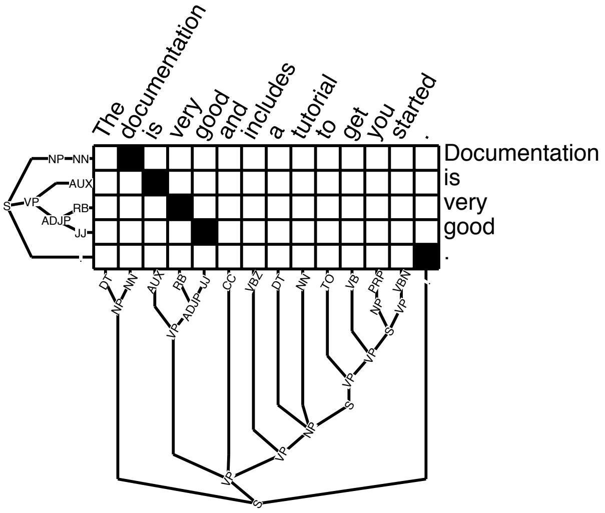
Our induction algorithm automatically finds grammar rules from a word-aligned parsed parallel corpus. The rules are pairs of *elementary trees* (i.e., tree fragments) whose leaf nodes are linked by the word alignments. These leaves can be either terminal or non-terminal symbols. Initially, the algorithm extracts tree pairs from word aligned text by choosing aligned constituents in the source and the target. These pairs are then generalised using subtrees which are also extracted, resulting in synchronous rules with variable nodes. The set of aligned tree pairs are extracted using the alignment template method (Och and Ney, 2004), constrained to syntactic constituent pairs:

$$\mathcal{C} = \{(n_S, n_T),\ (\exists(s,t) \in \mathcal{A} \wedge s \in Y(n_S) \wedge t \in Y(n_T)) \wedge \\ (\nexists(s,t) \in \mathcal{A} \wedge (s \in Y(n_S) \veebar t \in Y(n_T)))\}$$

where n_S and n_T are source and target tree nodes (subtrees), $\mathcal{A} = \{(s,t)\}$ is the set of word alignments (pairs of word-indices), $Y(\cdot)$ returns the yield span for a subtree and $\veebar$ is the exclusive-or operator.

The next step is to generalise the candidate pairs by replacing subtrees with variable nodes. We could fully trust the word alignments and adopt a strategy in which the rules are generalised as much as possible and thus include little lexicalisation. Figure 3 shows a simple sentence pair and the resulting synchronous rules according to this generalisation strategy. Alternatively, we could extract every possible rule by including unlexicalised rules, lexicalised rules and their combination. The downside here is that the total number of possible rules is factorial in the size of the candidate set. We address this problem by limiting the number of variables and the recursion depth, and by filtering out singleton rules.

There is no guarantee that the induced rules will generalise well to a testing set. For example, the testing data may have a rule which was not seen in the training set (*e.g.,* a new terminal or non terminal). In this case no rule can be applied and subsequently generation fails. For this reason we allow the model to duplicate any CFG production from the source tree, and uses a feature to flag that this rule was unseen in training. These SCFG rules are then merged with the induced rules and fed into the feature detection module (see Section 3.3 for details).



$$\begin{aligned}
\langle S,S\rangle &\rightarrow \langle [NP_{\boxed{1}}\, VP_{\boxed{2}}\ \cdot_{\boxed{3}}]_S, [NP_{\boxed{1}}\, VP_{\boxed{2}}\ \cdot_{\boxed{3}}]_S\rangle \\
\langle NP,NP\rangle &\rightarrow \langle [DT\ NN_{\boxed{1}}]_{NP}, [NN_{\boxed{1}}]_{NP}\rangle \\
\langle NN,NN\rangle &\rightarrow \langle \text{documentation}_{NN}, \text{Documentation}_{NN}\rangle \\
\langle VP,VP\rangle &\rightarrow \langle VP_{\boxed{1}}\, CC\ VP, VP_{\boxed{1}}\rangle \\
\langle VP,VP\rangle &\rightarrow \langle AUX_{\boxed{1}}\, ADJP_{\boxed{2}}, AUX_{\boxed{1}}\, ADJP_{\boxed{2}}\rangle \\
\langle AUX,AUX\rangle &\rightarrow \langle \text{is}_{AUX}, \text{is}_{AUX}\rangle \\
\langle ADJP,ADJP\rangle &\rightarrow \langle [RB_{\boxed{1}}\, JJ_{\boxed{2}}]_{ADJP}, [RB_{\boxed{1}}\, JJ_{\boxed{2}}]_{ADJP}\rangle \\
\langle RB,RB\rangle &\rightarrow \langle \text{very}_{RB}, \text{very}_{RB}\rangle \\
\langle JJ,JJ\rangle &\rightarrow \langle \text{good}_{ADJ}, \text{good}_{ADJ}\rangle \\
\langle .,.\rangle &\rightarrow \langle ..,..\rangle
\end{aligned}$$

Figure 3: Induced synchronous grammar from a sentence pair using a strategy that extracts general rules.

3.3 The Large Margin Model

We now describe how the parameters of our STSG generation system are fit to a supervised training set. For a given source tree, the space of sister target trees implied by the synchronous grammar is often very large, and the majority of these trees are ungrammatical or are poor compressions. The training procedure learns weights such that the model can discriminate between these trees and predict a good target tree. For this we develop a discriminative training process which learns a weighted tree-to-tree transducer. Our model is based on Tsochantaridis et al.'s (2005) framework for learning Support Vector Machines (SVMs) with structured output spaces, using the SVM^{struct} implementation.[1] We briefly summarise the approach below; for a more detailed description we refer the interested reader to Tsochantaridis et al. (2005).

Traditionally SVMs learn a linear classifier that separates two or more classes with the largest possible margin. Analogously, structured SVMs attempt to separate the correct structure from all other

[1] http://svmlight.joachims.org/svm_struct.html

structures with a large margin. Given an input instance **x**, we search for the optimum output **y** under the assumption that **x** and **y** can be adequately described using a combined feature vector representation $\Psi(\mathbf{x},\mathbf{y})$. Recall that **x** are the source trees and **y** are synchronous derivations which generate both **x** and a target tree.

$$f(\mathbf{x};\mathbf{w}) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\langle\mathbf{w},\Psi(\mathbf{x},\mathbf{y})\rangle \tag{2}$$

The goal of the training procedure is to find a parameter vector **w** such that it satisfies the condition:

$$\forall i, \forall y \in \mathcal{Y}\backslash y_i : \langle\mathbf{w},\Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\mathbf{y})\rangle \geq 0 \tag{3}$$

where $\mathbf{x}_i, \mathbf{y}_i$ are the ith training source tree and target derivation. To obtain a unique solution — there will be several parameter vectors **w** satisfying (3) if the training instances are linearly separable — Tsochantaridis et al. (2005) select the **w** that maximises the minimum distance between $\mathbf{y}_i$ and the closest runner-up structure.

The framework also incorporates a loss function. This property is particularly appealing in the context of sentence compression and generally text-to-text generation. For example, a compression that differs from the gold standard with respect to one or two words should be treated differently from a compression that bears no resemblance to it. Another important factor is the length of the compression. Compressions whose length is similar to the gold standard should be be preferable to longer or shorter output. A loss function $\Delta(\mathbf{y}_i,\mathbf{y})$ quantifies the accuracy of prediction **y** with respect to the true output value $\mathbf{y}_i$. We give details of the loss functions we employed for the compression task below.

We are now ready to state the learning objective for the structured SVM. We use the soft-margin formulation which allows errors in the training set, via the slack variables ξ_i:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i,\ \xi_i \geq 0 \tag{4}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y}\backslash \mathbf{y}_i : \langle\mathbf{w},\delta\Psi(\mathbf{y})\rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{y}_i,\mathbf{y})}$$

Slack variables ξ_i are introduced here for each training example x_i, C is a constant that controls the trade-off between training error minimisation and margin maximisation, and $\delta\Psi(\mathbf{y})$ is a shorthand for $\Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\mathbf{y})$ (see (3)). Note that slack variables are rescaled with the inverse loss incurred in each of the linear constraints.[2]

The optimisation problem in (4) is approximated using a polynomial time cutting plane algorithm (Tsochantaridis et al., 2005). This optimisation crucially relies on finding the constraint incurring the maximum cost. The cost function for slack rescaling can be formulated as:

$$H(\mathbf{y}) = (1 - \langle\delta\Psi_i(\mathbf{y}),\mathbf{w}\rangle)\Delta(\mathbf{y}_i,\mathbf{y}) \tag{5}$$

In order to adapt this framework to our generation problem, we must provide the feature mapping $\Psi(\mathbf{x},\mathbf{y})$, a loss function $\Delta(\mathbf{y}_i,\mathbf{y})$, and a maximiser $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y})$ (see (5)). The following sections describe how these are instantiated in the sentence compression task.

Feature Mapping We devised a general feature set suitable for compression and paraphrasing. Our feature space is defined over source trees (**x**) and target derivations (**y**). All features apply to a single grammar rule; a feature vector for a derivation is expressed as the sum of the feature vectors for each rule in this derivation.

We make use of syntactic, lexical, and compression specific features. Our simplest syntactic feature is the identity of a synchronous rule. Specifically, we record its source tree, its target tree and their combination. We also include rule frequencies $\phi(\mathit{target}|\mathit{source})$, $\phi(\mathit{source}|\mathit{target})$ and $\phi(\mathit{source},\mathit{target})$. Another feature records the frequencies of the CFG productions used in the target side of a rule. This allows the model to learn the weights of a CFG generation grammar, as a proxy for a language model. Using scores from a pre-trained CFG grammar or an n-gram language model might be preferable when the training sample is small, however we leave this as future work. Our last syntactic feature keeps track of the source root and the target root non-terminals. Our lexical features contain the list of tokens in the source yield, target yield, and both. We also use words as features.

[2] Alternatively, the loss function can be used to rescale the margin. This approach is less desirable as it is not scale invariant (Tsochantaridis et al., 2005). We also found empirically that slack-rescaling slightly outperforms margin rescaling on our compression task.

Finally, we have implemented a set of compression-specific features. These include a feature that detects if the yield of the target side of a synchronous rule is a subset of the yield of its source. We also take note of the edit operations (i.e., removal, insertion) required to transform the source side into the target. Edit operations are recorded separately for trees and their yields. In order to encourage compression, we also count the number of words on the target, the number of rules used in the derivation and the number of dropped variables.

Loss Functions The large margin configuration sketched above is quite modular and in theory a wide range of loss functions could be specified. Examples include edit-distance, precision, F-score, BLEU and tree-based measures. In practice, the loss function should be compatible with our maximisation algorithm which requires the objective function to decompose along the same lines as the tree derivation.[3]

Given this restriction, we define a loss based on position-independent unigram precision (Prec) which penalises errors in the yield independently for each word. Although fairly intuitive, this loss is far from ideal. First, it maximally rewards repeatedly predicting the same word if the latter is in the reference target tree. Secondly, it may bias towards overly short output which drops core information — one-word compressions will tend to have higher precision than longer output. To counteract this, we introduce two brevity penalty measures (BP) inspired by BLEU (Papineni et al., 2002) which we incorporate into the loss function, using a product, $loss = 1 - \text{Prec} \cdot \text{BP}$:

$$\begin{aligned} \text{BP1} &= \exp(1 - \max(1, \frac{r}{c})) \qquad (6) \\ \text{BP2} &= \exp(1 - \max(\frac{c}{r}, \frac{r}{c})) \end{aligned}$$

where r is the reference length and c is the candidate length.

BP1 is asymmetric, it has value one when $c \geq r$ and decays to zero when $c < r$. Note that precision should decay when $c > r$ as extra output will often not match the reference. BP2 is two-sided: it has value one when $c = r$ and decays towards zero for $c < r$ and $c > r$. In both cases, brevity is assessed against the gold standard target (not the source) to allow the system to learn the correct degree of compression from the training data.

Maximisation Algorithm Our algorithm finds the maximising derivation for $H(\mathbf{y})$ in (5). This derivation will have a high loss and a high score under the model, and therefore represents the most-violated constraint which is then added to the SVM's working set of constraints (see (4)).

The standard generation method from Section 3.1 cannot be used without modification to find the best scoring derivation since it does not account for the loss function or the gold standard derivation. Instead, we *stratify* the generation chart with the number of true and false positive tokens predicted, as described in Joachims (2005). These contingency values allow us to compute the precision and brevity penalty (see (6)) for each complete derivation. This is then combined with the derivation score and the gold standard derivation score to give $H(\mathbf{y})$.

The gold standard derivation features, $\Psi(\mathbf{x}_i, \mathbf{y}_i)$, must be calculated from a derivation linking the source tree to the gold target tree. As there may be many such derivations, we find a *unique* derivation using the smallest rules possible (for maximum generality). This is done using a dynamic program, similar to the inside-outside algorithm used in parsing. Other strategies are also possible, however we leave this to future work. Finally, we can find the global maximum $H(\mathbf{y})$ by maximising over all the root chart entries.

4 Evaluation Set-up

In this section we present our experimental set-up for assessing the performance of the max margin model described above. We give details of the corpora used, briefly introduce McDonald's (2006) sentence compression model used for comparison with our approach, and explain how system output was evaluated.

Corpora We evaluated our system on two different corpora. The first is the compression corpus of Knight and Marcu (2002) derived automatically from the document-abstract pairs of the Ziff-

[3] Optimising non-decompositional loss functions complicates the objective function, which then cannot be solved efficiently using a dynamic program.

Davis corpus. Previous compression work has almost exclusively used this corpus. Our experiments follow Knight and Marcu's partition of training, test, and development sets (1,002/36/12 instances). We also present results on Clarke and Lapata's (2006a) Broadcast News corpus.[4] This corpus was created manually (annotators were asked to produce compressions for 50 Broadcast news stories) and poses more of a challenge than Ziff-Davis. Being a speech corpus, it often contains incomplete and ungrammatical utterances and speech artefacts such as disfluencies, false starts and hesitations. Furthermore, spoken utterances have varying lengths, some are very wordy whereas others cannot be reduced any further. Thus a hypothetical compression system trained on this domain should be able to leave some sentences uncompressed. Again we used Clarke and Lapata's training, test, and development set split (882/410/78 instances).

Comparison with State-of-the-art We evaluated our approach against McDonald's (2006) discriminative model. This model is a good basis for comparison for several reasons. First, it achieves competitive performance with Knight and Marcu's (2002) decision tree and noisy channel models. Second, it also uses large margin learning. Sentence compression is formulated as a string-to-substring mapping problem with a deletion-based Hamming loss. Recall that our formulation involves a tree-to-tree mapping. Third, it uses a feature space complementary to ours. For example features are defined between adjacent words, and syntactic evidence is incorporated indirectly into the model. In contrast our model relies on synchronous rules to generate valid compressions and does not explicitly incorporate adjacency features. We used an implementation of McDonald (2006) for comparison of results (Clarke and Lapata, 2007).

Evaluation Measures In line with previous work we assessed our model's output by eliciting human judgements. Participants were presented with an original sentence and its compression and asked to rate the latter on a five point scale based on the information retained and its grammaticality. We conducted two separate elicitation studies, one for the Ziff-Davis and one for the Broadcast news dataset. In both cases our materials consisted of 96 source-target sentences. These included gold standard compressions and the output of our system and McDonald's (2006). We were able to obtain ratings on the entire Ziff-Davis test set as it has only 32 instances; this was not possible for Broadcast news as the test section consists of 410 instances. Consequently, we randomly selected 32 source-target sentences to match the size of the Ziff-Davis test set.[5] We collected ratings from 60 unpaid volunteers, all self reported native English speakers. Both studies were conducted over the Internet. Examples of our experimental items are given in Table 1.

O:	I just wish my parents and my other teachers could be like this teacher, so we could communicate.
M:	I wish my teachers could be like this teacher.
S:	I wish my teachers could be like this, so we could communicate.
G:	I wish my parents and other teachers could be like this, so we could communicate.
O:	Earlier this week, in a conference call with analysts, the bank said it boosted credit card reserves by $350 million.
M:	Earlier said credit card reserves by $350 million.
S:	In a conference call with analysts, the bank boosted card reserves by $350 million.
G:	In a conference call with analysts the bank said it boosted credit card reserves by $350 million.

Table 1: Compression examples from the Broadcast news corpus (O: original sentence, M: McDonald (2006), S: STSG, G: gold standard)

We also report results using F1 computed over grammatical relations (Riezler et al., 2003). We chose F1 (as opposed to accuracy or edit distance-based measures) as Clarke and Lapata (2006b) show that it correlates reliably with human judgements.

5 Experiments

The framework presented in Section 3 is quite flexible. Depending on the grammar induction strategy, choice of features, loss function and maximisation algorithm, different classes of models can be derived. Before presenting our results in detail we discuss the specific model employed in our experiments and explain how its parameters were instantiated.

In order to build a compression model we need

[4] The corpus can be downloaded from `http://homepages.inf.ed.ac.uk/s0460084/data/`.

[5] A Latin square design ensured that subjects did not see two different compressions of the same sentence.

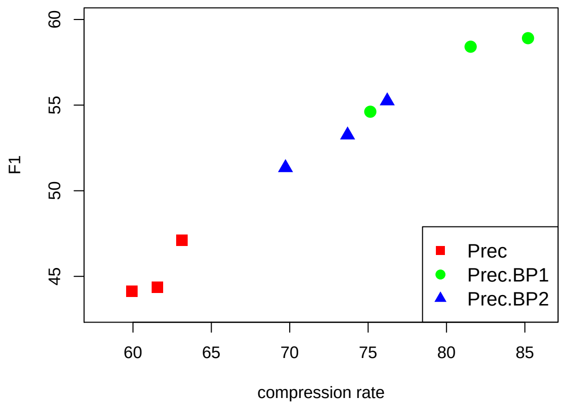


Figure 4: Compression rate vs. grammatical relations F1 using unigram precision alone and in combination with two brevity penalties.

a parallel corpus of syntax trees. We obtained syntactic analyses for source and target sentences with Bikel's (2002) parser. Our corpora were automatically aligned with Giza++ (Och et al., 1999) in both directions between source and target and symmetrised using the intersection heuristic (Koehn et al., 2003). Each word in the lexicon was also aligned with itself. This was necessary in order to inform Giza++ about word identity. Unparseable sentences and those longer than 50 tokens were removed from the data set.

We induced a synchronous tree substitution grammar from the Ziff-Davis and Broadcast news corpora using the method described in Section 3.2. We extracted all maximally general synchronous rules. These were complemented with more specific rules from conjoining pairs of general rules. The specific rules were pruned to remove singletons and those rules with more than 3 variables. Grammar rules were represented by the features described in Section 3.3.

An important parameter for our compression task is the appropriate choice of loss function. Ideally, we would like a loss function that encourages compression without overly aggressive information loss. Figure 4 plots compression rate against grammatical relations F1 using each of the loss functions presented in Section 3.3 on the Ziff-Davis development set.[6] As can be seen with unigram precision alone (Prec) the system produces overly short output, whereas the one-sided brevity penalty (BP1) achieves the opposite effect. The two-sided brevity penalty (BP2) seems to strike the right balance: it encourages compression while achieving good F-scores. This suggests that important information is retained in spite of significant compression. We also varied the regularisation parameter C (see (4)) over a range of values on the development set and found that setting it to 0.01 yields overall good performance across corpora and loss functions.

Ziff-Davis	CompR	RelF1
McDonald06	66.2	45.8
STSG	56.8	54.3
Gold standard	57.2	—

Broadcast News	CompR	RelF1
McDonald06	68.6	47.6
STSG	73.7	53.4*
Gold standard	76.1	—

Table 2: Results using grammatical relations F1 (*: sig. diff. from McDonald06; $p < 0.01$ using the Student t test)

We now present our results on the test set. These were obtained with a model that uses slack rescaling and a precision-based loss function with a two-sided brevity penalty ($C = 0.01$). Table 2 shows the average compression rates (CompR) for McDonald (2006) and our model (STSG) as well as their performance according to grammatical relations F1. The row 'Gold standard' displays human-produced compression rates. Notice that our model obtains compression rates similar to the gold standard, whereas McDonald tends to compress less on Ziff-Davis and more on Broadcast news. As far as F1 is concerned, we see that STSG outperforms McDonald on both corpora. The difference in F1 is statistically significant on Broadcast news but not on Ziff-Davis (which consists solely of 32 sentences).

Table 3 presents the results of our elicitation study. We carried out an Analysis of Variance (ANOVA) to examine the effect of system type (McDonald06, STSG, Gold standard) on the compression ratings. The ANOVA revealed a reliable effect on both corpora. We used post-hoc Tukey tests to

[6] We obtained a similar plot for the Broadcast News corpus but omit it due to lack of space.

Model	Ziff-Davis	Broadcast news
McDonald06	2.82[†]	2.16[†]
STSG	3.20[†*]	2.63[*]
Gold standard	3.72	3.05

Table 3: Mean ratings on compression output elicited by humans ([*]: sig. diff. from McDonald06 ($\alpha < 0.05$); [†] sig. diff. from Gold standard ($\alpha < 0.01$); using post-hoc Tukey tests)

examine whether the mean ratings for each system differed significantly. The Tukey tests showed that STSG is perceived as significantly better than McDonald06. There is no significant difference between STSG and the gold standard compressions on the Broadcast news; both systems are significantly worse than the gold standard on Ziff-Davis.

These results are encouraging, indicating that our highly expressive framework is a good model for sentence compression. Under several experimental conditions we obtain better performance than previous work. Importantly, the model described here is not compression-specific, it could be easily adapted to other tasks, corpora or languages (for which syntactic analysis tools are available). Being supervised, our model learns to fit the compression rate of the training data. In this sense, it is somewhat inflexible as it cannot easily adapt to a specific rate given by a user or imposed by an application (*e.g.,* when displaying text on small screens). Compression rate can be indirectly manipulated by adopting loss functions that encourage or discourage compression (see Figure 4), but admittedly in other frameworks (*e.g.,* Clarke and Lapata (2006a)) the length of the compression can be influenced more naturally.

In our formulation of the compression problem, a derivation is characterised by a single inventory of features. This entails that the feature space cannot in principle distinguish between derivations that use the same rules, applied in a different order. Although, this situation does not arise often in our dataset, we believe that it can be ameliorated by intersecting a language model with our generation algorithm (Chiang, 2005).

6 Conclusions and Future Work

In this paper we have presented a novel method for sentence compression cast in the framework of structured learning. We develop a system that generates compressions using a synchronous tree substitution grammar whose weights are discriminatively trained within a large margin model. We also describe an appropriate algorithm than can be used in both training (i.e., learning the model weights) and decoding (i.e., finding the most plausible compression under the model). The proposed formulation allows us to capture rewriting operations that go beyond word deletion and can be easily tuned to specific loss functions directly related to the problem at hand. We empirically evaluate our approach against a state-of-the art model (McDonald, 2006) and show performance gains on two compression corpora.

Future research will follow three directions. First, we will extend the framework to incorporate position dependent loss functions. Examples include the Hamming distance or more sophisticated functions that take the tree structure of the source and target sentences into account. Such functions can be supported by augmenting our generation algorithm with a beam search. Secondly, the present paper used a relatively simple feature set. Our intention was to examine our model's performance without extensive feature engineering. Nevertheless, improvements should be possible by incorporating features defined over *n*-grams and dependencies (McDonald, 2006). Finally, the experiments presented in this work use a grammar acquired from the training corpus. However, there is nothing inherent in our formalisation that restricts us to this particular grammar. We therefore plan to investigate the potential of our method with unsupervised or semi-supervised grammar induction techniques for additional rewriting tasks including paraphrase generation and machine translation.

Acknowledgements The authors acknowledge the support of EPSRC (grants GR/T04540/01 and GR/T04557/01). We are grateful to James Clarke for sharing his implementation of McDonald (2006) with us. Special thanks to Philip Blunsom for insightful comments and suggestions.

References

R. Barzilay, K. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL/EACL*, 50–57, Toulouse, France.

D. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT*, 24–27, San Diego, CA.

J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, J. Tait. 1999. Simplifying text for language impaired readers. In *Proceedings of EACL*, 269–270, Bergen, Norway.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd ACL*, 263–270, Ann Arbor, MI.

J. Clarke, M. Lapata. 2006a. Constraint-based sentence compression: An integer programming approach. In *Proceedings of COLING/ACL Main Conference Poster Sessions*, 144–151, Sydney, Australia.

J. Clarke, M. Lapata. 2006b. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of COLING/ACL*, 377–384, Sydney, Australia.

J. Clarke, M. Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of EMNLP-CoNLL*, Prague, Czech Republic.

J. Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the ACL Interactive Poster/Demonstration Sessions*, 205–208, Sapporo, Japan.

M. Galley, K. McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Proceedings of NAACL/HLT*, 180–187, Rochester, NY.

J. Grael, K. Knight. 2004. Training tree transducers. In *Proceedings of NAACL/HLT*, 105–112, Boston, MA.

C. Hori, S. Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, E87-D(1):15–25.

H. Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of ANLP*, 310–315, Seattle, WA.

T. Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of ICML*, 377–384, Bonn, Germany.

K. Knight, D. Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

P. Koehn, F. J. Och, D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*, 48–54, Edmonton, Canada.

D. Lin, P. Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):342–360.

R. McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proceedings of EACL*, 297–304, Trento, Italy.

I. D. Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of ACL*, 653–660, Barcelona, Spain.

F. J. Och, H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. J. Och, C. Tillmann, H. Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of EMNLP/VLC*, 20–28, College Park, MD.

B. Pang, K. Knight, D. Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of NAACL*, 181–188, Edmonton, Canada.

K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, 311–318, Philadelphia, PA.

C. Quirk, C. Brockett, W. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*, 142–149, Barcelona, Spain.

S. Riezler, T. H. King, R. Crouch, A. Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of HLT/NAACL*, 118–125, Edmonton, Canada.

Y. Shinyama, S. Sekine, K. Sudo, R. Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT*, 40–46, San Diego, CA.

I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

J. Turner, E. Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*, 290–297, Ann Arbor, MI.

L. S. Zettlemoyer, M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*, 825–830, Edinburgh, UK.

Incremental Text Structuring with Online Hierarchical Ranking

Erdong Chen, Benjamin Snyder and Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{edc,bsnyder,regina}@csail.mit.edu

Abstract

Many emerging applications require documents to be repeatedly updated. Such documents include newsfeeds, webpages, and shared community resources such as Wikipedia. In this paper we address the task of inserting new information into existing texts. In particular, we wish to determine the best location in a text for a given piece of new information. For this process to succeed, the insertion algorithm should be informed by the existing document structure. Lengthy real-world texts are often hierarchically organized into chapters, sections, and paragraphs. We present an online ranking model which exploits this hierarchical structure – representationally in its features and algorithmically in its learning procedure. When tested on a corpus of Wikipedia articles, our hierarchically informed model predicts the correct insertion paragraph more accurately than baseline methods.

1 Introduction

Many emerging applications require documents to be repeatedly updated. For instance, newsfeed articles are continuously revised by editors as new information emerges, and personal webpages are modified as the status of the individual changes. This revision strategy has become even more prevalent with the advent of community edited web resources, the most notable example being Wikipedia. At present this process involves massive human effort. For instance, the English language version of Wikipedia averaged over 3 million edits[1] per month in 2006. Even so, many articles quickly become outdated. A system that performs such updates automatically could drastically decrease maintenance efforts and potentially improve document quality.

Currently there is no effective way to automatically update documents as new information becomes available. The closest relevant text structuring technique is the work on sentence ordering, in which a complete reordering of the text is undertaken. Predictably these methods are suboptimal for this new task because they cannot take advantage of existing text structure.

We introduce an alternative vision of text structuring as a process unfolding over time. Instead of ordering sentences all at once, we start with a well-formed draft and add new information at each stage, while preserving document coherence. The basic operation of incremental text structuring is the insertion of new information. To automate this process, we develop a method for determining the best location in a text for a given piece of new information.

The main challenge is to maintain the continuity and coherence of the original text. These properties may be maintained by examining sentences adjacent to each potential insertion point. However, a local sentence comparison method such as this may fail to account for global document coherence (e.g. by allowing the mention of some fact in an inappropriate section). This problem is especially acute in the case of lengthy, real-world texts such as books, technical reports, and web pages. These documents

[1] http://stats.wikimedia.org/EN/TablesWikipediaEN.htm

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 83–91, Prague, June 2007. ©2007 Association for Computational Linguistics

are commonly organized hierarchically into sections and paragraphs to aid reader comprehension. For documents where hierarchical information is not explicitly provided, such as automatic speech transcripts, we can use automatic segmentation methods to induce such a structure (Hearst, 1994). Rather than ignoring the inherent hierarchical structure of these texts, we desire to directly model such hierarchies and use them to our advantage – both representationally in our features and algorithmically in our learning procedure.

To achieve this goal, we introduce a novel method for sentence insertion that operates over a hierarchical structure. Our document representation includes features for each layer of the hierarchy. For example, the word overlap between the inserted sentence and a section header would be included as an upper-level section feature, whereas a comparison of the sentence with all the words in a paragraph would be a lower-level paragraph feature. We propose a linear model which simultaneously considers the features of every layer when making insertion decisions. We develop a novel update mechanism in the online learning framework which exploits the hierarchical decomposition of features. This mechanism limits model updates to those features found at the highest incorrectly predicted layer, without unnecessarily disturbing the parameter values for the lower reaches of the tree. This conservative update approach maintains as much knowledge as possible from previously encountered training examples.

We evaluate our method using real-world data where multiple authors have revised preexisting documents over time. We obtain such a corpus from Wikipedia articles,[2] which are continuously updated by multiple authors. Logs of these updates are publicly available, and are used for training and testing of our algorithm. Figure 1 shows an example of a Wikipedia insertion. We believe this data will more closely mirror potential applications than synthetic collections used in previous work on text structuring.

Our hierarchical training method yields significant improvement when compared to a similar non-hierarchical model which instead uses the standard perceptron update of Collins (2002). We also report human performance on the insertion task in order to provide a reasonable upper-bound on machine performance. An analysis of these results shows that our method closes the gap between machine and human performance substantially.

In the following section, we provide an overview of existing work on text structuring and hierarchical learning. Then, we define the insertion task and introduce our hierarchical ranking approach to sentence insertion. Next, we present our experimental framework and data. We conclude the paper by presenting and discussing our results.

2 Related Work

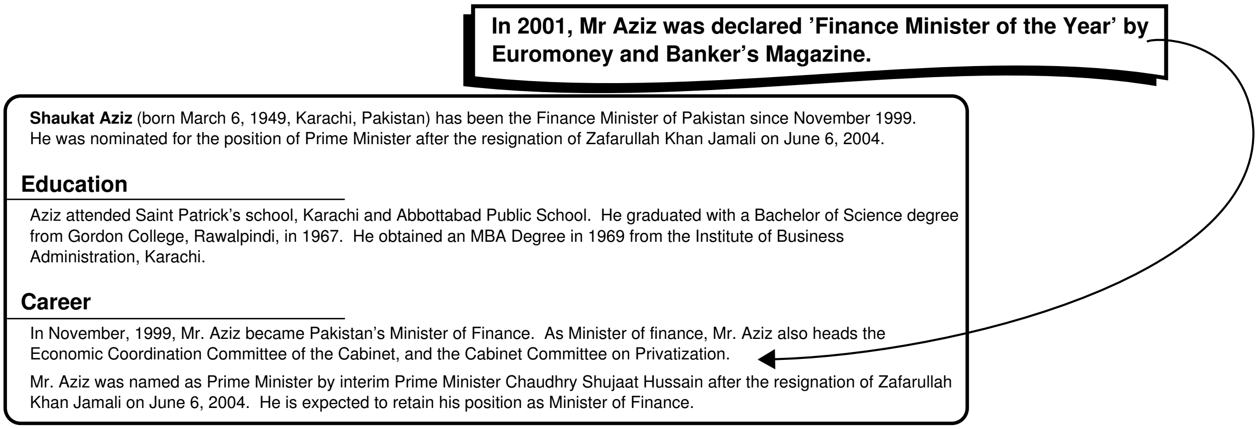
Text Structuring The insertion task is closely related to the extensively studied problem of sentence ordering.[3] Most of the existing algorithms represent text structure as a linear sequence and are driven by local coherence constraints (Lapata, 2003; Karamanis et al., 2004; Okazaki et al., 2004; Barzilay and Lapata, 2005; Bollegala et al., 2006; Elsner and Charniak, 2007). These methods induce a total ordering based on pairwise relations between sentences. Researchers have shown that identifying precedence relations does not require deep semantic interpretation of input sentences: shallow distributional features are sufficient for accurate prediction. Our approach employs similar features to represent nodes at the lowest level of the hierarchy.

The key departure of our work from previous research is the incorporation of hierarchical structure into a corpus-based approach to ordering. While in symbolic generation and discourse analysis a text is typically analyzed as a tree-like structure (Reiter and Dale, 1990), a linear view is prevalent in data-driven methods to text structuring.[4] Moving beyond a linear representation enables us to handle longer texts where a local view of coherence does not suffice. At the same time, our approach does not require any manual rules for handling tree insertions, in contrast to symbolic text planners.

[2] Data and code used in this paper are available at `http://people.csail.mit.edu/edc/emnlp07/`

[3] Independently and simultaneously with our work, Elsner and Charniak (2007) have studied the sentence insertion task in a different setting.

[4] Though statistical methods have been used to induce such trees (Soricut and Marcu, 2003), they are not used for ordering and other text-structuring tasks.



In 2001, Mr Aziz was declared ’Finance Minister of the Year’ by Euromoney and Banker’s Magazine.

Shaukat Aziz (born March 6, 1949, Karachi, Pakistan) has been the Finance Minister of Pakistan since November 1999. He was nominated for the position of Prime Minister after the resignation of Zafarullah Khan Jamali on June 6, 2004.

Education

Aziz attended Saint Patrick’s school, Karachi and Abbottabad Public School. He graduated with a Bachelor of Science degree from Gordon College, Rawalpindi, in 1967. He obtained an MBA Degree in 1969 from the Institute of Business Administration, Karachi.

Career

In November, 1999, Mr. Aziz became Pakistan’s Minister of Finance. As Minister of finance, Mr. Aziz also heads the Economic Coordination Committee of the Cabinet, and the Cabinet Committee on Privatization.

Mr. Aziz was named as Prime Minister by interim Prime Minister Chaudhry Shujaat Hussain after the resignation of Zafarullah Khan Jamali on June 6, 2004. He is expected to retain his position as Minister of Finance.

Figure 1: An example of Wikipedia insertion.

Hierarchical Learning There has been much recent research on multiclass hierarchical classification. In this line of work, the set of possible labels is organized hierarchically, and each input must be assigned a node in the resulting tree. A prototype weight vector is learned for each node, and classification decisions are based on all the weights along the path from node to root. The essence of this scheme is that the more ancestors two nodes have in common, the more parameters they are forced to share. Many learning methods have been proposed, including SVM-style optimization (Cai and Hofmann, 2004), incremental least squares estimation (Cesa-Bianchi et al., 2006b), and perceptron (Dekel et al., 2004).

This previous work rests on the assumption that a predetermined set of atomic labels with a fixed hierarchy is given. In our task, however, the set of possible insertion points – along with their hierarchical organization – is unique to each input document. Furthermore, nodes exhibit rich internal feature structure and cannot be identified across documents, except insofar as their features overlap. As is commonly done in NLP tasks, we make use of a feature function which produces one feature vector for each possible insertion point. We then choose among these feature vectors using a single weight vector (casting the task as a *structured ranking* problem rather than a *classification* problem). In this framework, an explicit hierarchical view is no longer necessary to achieve parameter tying. In fact, each parameter will be shared by exactly those insertion points which exhibit the corresponding feature, both across documents and within a single document. Higher level parameters will thus naturally be shared by all paragraphs within a single section.

In fact, when the perceptron update rule of (Dekel et al., 2004) – which modifies the weights of every divergent node along the predicted and true paths – is used in the ranking framework, it becomes virtually identical with the standard, flat, ranking perceptron of Collins (2002).[5] In contrast, our approach shares the idea of (Cesa-Bianchi et al., 2006a) that “if a parent class has been predicted wrongly, then errors in the children should not be taken into account.” We also view this as one of the key ideas of the incremental perceptron algorithm of (Collins and Roark, 2004), which searches through a complex decision space step-by-step and is immediately updated at the first wrong move.

Our work fuses this idea of selective hierarchical updates with the simplicity of the perceptron algorithm and the flexibility of arbitrary feature sharing inherent in the ranking framework.

3 The Algorithm

In this section, we present our sentence insertion model and a method for parameter estimation. Given a hierarchically structured text composed of sections and paragraphs, the sentence insertion model determines the best paragraph within

[5] The main remaining difference is that Dekel et al. (2004) use a passive-aggressive update rule (Crammer et al., 2006) and in doing so enforce a margin based on tree distance.

which to place the new sentence. To identify the exact location of the sentence within the chosen paragraph, local ordering methods such as (Lapata, 2003) could be used. We formalize the insertion task as a structured ranking problem, and our model is trained using an online algorithm. The distinguishing feature of the algorithm is a selective correction mechanism that focuses the model update on the relevant layer of the document's feature hierarchy.

The algorithm described below can be applied to any hierarchical ranking problem. For concreteness, we use the terminology of the sentence insertion task, where a hierarchy corresponds to a document with sections and paragraphs.

3.1 Problem Formulation

In a sentence insertion problem, we are given a training sequence of instances $(s^1, \mathcal{T}^1, \ell^1), \ldots, (s^m, \mathcal{T}^m, \ell^m)$. Each instance contains a sentence s, a hierarchically structured document $\mathcal{T}$, and a node ℓ representing the correct insertion point of s into $\mathcal{T}$. Although ℓ can generally be any node in the tree, in our problem we need only consider leaf nodes. We cast this problem in the ranking framework, where a feature vector is associated with each sentence-node pair. For example, the feature vector of an internal, section-level node may consider the word overlap between the inserted sentence and the section title. At the leaf level, features may include an analysis of the overlap between the corresponding text and sentence. In practice, we use disjoint feature sets for different layers of the hierarchy, though in theory they could be shared.

Our goal then is to choose a leaf node by taking into account its feature vector as well as feature vectors of all its ancestors in the tree.

More formally, for each sentence s and hierarchically structured document $\mathcal{T}$, we are given a set of feature vectors, with one for each node: $\{\phi(s,n) : n \in \mathcal{T}\}$. We denote the set of leaf nodes by $\mathcal{L}(\mathcal{T})$ and the path from the root of the tree to a node n by $\mathcal{P}(n)$. Our model must choose one leaf node among the set $\mathcal{L}(\mathcal{T})$ by examining its feature vector $\phi(s,\ell)$ as well as all the feature vectors along its path: $\{\phi(s,n) : n \in \mathcal{P}(\ell)\}$.

```
Input : $(s^1, \mathcal{T}^1, \ell^1), \ldots, (s^m, \mathcal{T}^m, \ell^m)$.
Initialize : Set $\mathbf{w}^1 = 0$
Loop : For $t = 1, 2, ..., N$ :
  1. Get a new instance $s^t, \mathcal{T}^t$.
  2. Predict $\hat{\ell}^t = \arg\max_{\ell \in \mathcal{L}(\mathcal{T})} \mathbf{w}^t \cdot \Phi(s^t, \ell)$.
  3. Get the new label $\ell^t$.
  4. If $\hat{\ell}^t = \ell^t$:
       $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t$
     Else:
       $i^* \leftarrow \max\{i : \mathcal{P}(\ell^t)^i = \mathcal{P}(\hat{\ell}^t)^i\}$
       $a \leftarrow \mathcal{P}(\ell^t)^{i^*+1}$
       $b \leftarrow \mathcal{P}(\hat{\ell}^t)^{i^*+1}$
       $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \phi(s,a) - \phi(s,b)$
Output : $\mathbf{w}^{N+1}$.
```

Figure 2: Training algorithm for the hierarchical ranking model.

3.2 The Model

Our model consists of a weight vector $\mathbf{w}$, each weight corresponding to a single feature. The features of a leaf are aggregated with the features of all its ancestors in the tree. The leaf score is then computed by taking the inner product of this aggregate feature vector with the weights $\mathbf{w}$. The leaf with the highest score is then selected.

More specifically, we define the *aggregate feature vector* of a leaf ℓ to be the sum of all features found along the path to the root:

$$\Phi(s,\ell) = \sum_{n \in \mathcal{P}(\ell)} \phi(s,n) \tag{1}$$

This has the effect of *stacking together* features found in a single layer, and *adding* the values of features found at more than one layer.

Our model then outputs the leaf with the highest scoring aggregate feature vector:

$$\arg\max_{\ell \in \mathcal{L}(\mathcal{T})} \mathbf{w} \cdot \Phi(s,\ell) \tag{2}$$

Note that by using this criterion, our decoding method is equivalent to that of the standard linear ranking model. The novelty of our approach lies in our training algorithm which uses the hierarchical feature decomposition of Equation 1 to pinpoint its updates along the path in the tree.

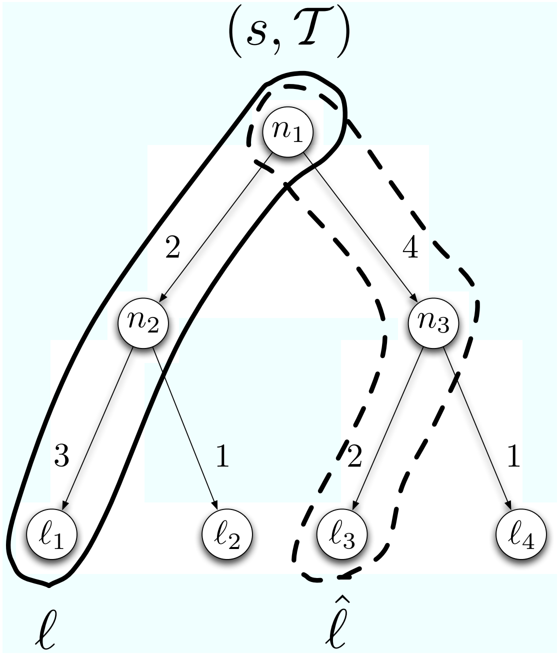


Figure 3: An example of a tree with the corresponding model scores. The path surrounded by solid lines leads to the correct node ℓ_1. The path surrounded by dotted lines leads to ℓ_3, the predicted output based on the current model.

3.3 Training

Our training procedure is implemented in the online learning framework. The model receives each training instance, and predicts a leaf node according to its current parameters. If an incorrect leaf node is predicted, the weights are updated based on the divergence between the predicted path and the true path. We trace the paths down the tree, and only update the weights of the features found at the split point. Updates for shared nodes along the paths would of course cancel out. In contrast to the standard ranking perceptron as well as the hierarchical perceptron of (Dekel et al., 2004), no features further down the divergent paths are incorporated in the update. For example, if the model incorrectly predicts the section, then only the weights of the section features are updated whereas the paragraph feature weights remain untouched.

More formally, let $\hat{\ell}$ be the predicted leaf node and let $\ell \neq \hat{\ell}$ be the true leaf node. Denote by $\mathcal{P}(\ell)^i$ the i^{th} node on the path from the root to ℓ. Let i^* be the depth of the lowest common ancestor of ℓ and $\hat{\ell}$ (i.e., $i^* = \max\{i : \mathcal{P}(\ell)^i = \mathcal{P}(\hat{\ell})^i\}$). Then the update rule for this round is:

$$\mathbf{w} \leftarrow \mathbf{w} + \phi\left(s, \mathcal{P}(\ell)^{i^*+1}\right) - \phi\left(s, \mathcal{P}(\hat{\ell})^{i^*+1}\right) \quad (3)$$

Full pseudo-code for our hierarchical online training algorithm is shown in Figure 2.

We illustrate the selective update mechanism on the simple example shown on Figure 3. The correct prediction is the node ℓ_1 with an aggregate path score of 5, but ℓ_3 with the higher score of 6 is predicted. In this case, both the section and the paragraph are incorrectly predicted. In response to this mistake, the features associated with the correct section, n_2, are added to the weights, and the features of the incorrectly predicted section, n_3, are subtracted from the weights. An alternative update strategy would be to continue to update the feature weights of the leaf nodes, ℓ_1 and ℓ_3. However, by identifying the exact source of path divergence we preserve the previously learned balance between leaf node features.

4 Features

Features used in our experiments are inspired by previous work on corpus-based approaches for discourse analysis (Marcu and Echihabi, 2002; Lapata, 2003; Elsner et al., 2007). We consider three types of features: lexical, positional, and temporal. This section gives a general overview of these features (see code for further details.)

Lexical Features Lexical features have been shown to provide strong cues for sentence positioning. To preserve text cohesion, an inserted sentence has to be topically close to its surrounding sentences. At the paragraph level, we measure topical overlap using the TF*IDF weighted cosine similarity between an inserted sentence and a paragraph. We also use a more linguistically refined similarity measure that computes overlap considering only subjects and objects. Syntactic analysis is performed using the MINIPAR parser (Lin, 1998).

The overlap features are computed at the section level in a similar way. We also introduce an additional section-level overlap feature that computes the cosine similarity between an inserted sentence and the first sentence in a section. In our corpus, the opening sentence of a section is typically strongly indicative of its topic, thus providing valuable cues for section level insertions.

In addition to overlap, we use lexical features that capture word co-occurrence patterns in coherent texts. This measure was first introduced in the context of sentence ordering by Lapata (2003). Given a collection of documents in a specific domain, we compute the likelihood that a pair of words co-occur in adjacent sentences. From these counts, we induce the likelihood that two sentences are adjacent to each other. For a given paragraph and an inserted sentence, the highest adjacency probability between the inserted sentence and paragraph sentences is recorded. This feature is also computed at the section level.

Positional Features These features aim to capture user preferences when positioning new information into the body of a document. For instance, in the Wikipedia data, insertions are more likely to appear at the end of a document than at its beginning. We track positional information at the section and paragraph level. At the section level, we record whether a section is the first or last of the document. At the paragraph level, there are four positional features which indicate the paragraph's position (i.e., start or end) within its individual section and within the document as a whole.

Temporal Features The text organization may be influenced by temporal relations between underlying events. In temporally coherent text, events that happen in the same time frame are likely to be described in the same segment. Our computation of temporal features does not require full fledged temporal interpretation. Instead, we extract these features based on two categories of temporal cues: verb tense and date information. The verb tense feature captures whether a paragraph contains at least one sentence using the same tense as the inserted sentence. For instance, this feature would occur for the inserted sentence in Figure 1 since both the sentence and chosen paragraph employ the past tense.

Another set of features takes into account the relation between the dates in a paragraph and those in an inserted sentence. We extract temporal expressions using the **TIMEX2** tagger (Mani and Wilson, 2000), and compute the time interval for a paragraph bounded by its earliest and latest dates. We record the degree of overlap between the paragraph time interval and insertion sentence time interval.

		Section	Paragraph	Tree Dist
T1	J1	0.575	0.5	1.85
	J2	0.7	0.525	1.55
T2	J3	0.675	0.55	1.55
	J4	0.725	0.55	1.45

Table 1: Accuracy of human insertions compared against gold standard from Wikipedia's update log. T1 is a subset of the data annotated by judges J1 and J2, while T2 is annotated by J3 and J4.

5 Experimental Set-Up

Corpus Our corpus consists of Wikipedia articles that belong to the category "Living People." We focus on this category because these articles are commonly updated: when new facts about a person are featured in the media, a corresponding entry in Wikipedia is likely to be modified. Unlike entries in a professionally edited encyclopedia, these articles are collaboratively written by multiple users, resulting in significant stylistic and content variations across texts in our corpus. This property distinguishes our corpus from more stylistically homogeneous collections of biographies used in text generation research (Duboue and McKeown, 2003).

We obtain data on insertions[6] from the update log that accompanies every Wikipedia entry. For each change in the article's history, the log records an article before and after the change. From this information, we can identify the location of every inserted sentence. In cases where multiple insertions occur over time to the same article, they are treated independently of each other. To eliminate spam, we place constraints on inserted sentences: (1) a sentence has at least 8 tokens and at most 120 tokens; (2) the MINIPAR parser (Lin, 1998) can identify a subject or an object in a sentence.

This process yields 4051 insertion/article pairs, from which 3240 pairs are used for training and 811 pairs for testing. These insertions are derived from 1503 Wikipedia articles. Relative to other corpora used in text structuring research (Barzilay and Lee, 2004; Lapata, 2003; Karamanis et al., 2004), texts in

[6] Insertion is only one type of recorded update, others include deletions and sentence rewriting.

our collection are long: an average article has 32.9 sentences, organized in 3.61 sections and 10.9 paragraphs. Our corpus only includes articles that have more than one section. When sentences are inserted between paragraphs, by convention we treat them as part of the previous paragraph.

Evaluation Measures We evaluate our model using *insertion accuracy* at the section and paragraph level. This measure computes the percentage of matches between the predicted location of the insertion and the true placement. We also report the *tree distance* between the predicted position and the true location of an inserted sentence. *Tree distance* is defined as the length of the path through the tree which connects the predicted and the true paragraph positions. This measure captures section level errors (which raise the connecting path higher up the tree) as well as paragraph level errors (which widen the path across the tree).

Baselines Our first three baselines correspond to naive insertion strategies. The RANDOMINS method randomly selects a paragraph for a new sentence, while FIRSTINS and LASTINS insert a sentence into the first and the last paragraph, respectively.

We also compare our HIERARCHICAL method against two competitive baselines, PIPELINE and FLAT. The PIPELINE method separately trains two rankers, one for section selection and one for paragraph selection. During decoding, the PIPELINE method first chooses the best section according to the section-layer ranker, and then selects the best paragraph within the chosen section according to the paragraph-layer ranker. The FLAT method uses the same decoding criterion as our model (Equation 2), thus making use of all the same features. However, FLAT is trained with the standard ranking perceptron update, without making use of the hierarchical decomposition of features in Equation 1.

Human Performance To estimate the difficulty of sentence insertion, we conducted experiments that evaluate human performance on the task. Four judges collectively processed 80 sentence/article pairs which were randomly extracted from the test set. Each insertion was processed by two annotators.

Table 1 shows the insertion accuracy for each judge when compared against the Wikipedia gold standard. On average, the annotators achieve 66% accuracy in section placement and 53% accuracy in paragraph placement. We obtain similar results when we compare the agreement of the judges against each other: 65% of section inserts and 48% of paragraph inserts are identical between two annotators. The degree of variability observed in this experiment is consistent with human performance on other text structuring tasks such as sentence ordering (Barzilay et al., 2002; Lapata, 2003).

	Section	Paragraph	Tree Dist
RANDOMINS	0.318*	0.134*	3.10*
FIRSTINS	0.250*	0.136*	3.23*
LASTINS	0.305*	0.215*	2.96*
PIPELINE	0.579	0.314*	2.21*
FLAT	0.593	0.313*	2.19*
HIERARCHY	**0.598**	**0.383**	**2.04**

Table 2: Accuracy of automatic insertion methods compared against the gold standard from Wikipedia's update log. The third column gives tree distance, where a lower score corresponds to better performance. Diacritic * ($p < 0.01$) indicates whether differences in accuracy between the given model and the Hierarchical model is significant (using a Fisher Sign Test).

6 Results

Table 2 shows the insertion performance of our model and the baselines in terms of accuracy and tree distance error. The two evaluation measures are consistent in that they yield roughly identical rankings of the systems. Assessment of statistical significance is performed using a Fisher Sign Test. We apply this test to compare the accuracy of the HIERARCHICAL model against each of the baselines.

The results in Table 2 indicate that the naive insertion baselines (RANDOMINS, FIRSTINS, LASTINS) fall substantially behind the more sophisticated, trainable strategies (PIPELINE, FLAT, HIERARCHICAL). Within the latter group, our HIERARCHICAL model slightly outperforms the others based on the coarse measure of accuracy at the section level. However, in the final paragraph-level analysis, the performance gain of our model over its counterparts is quite significant. Moreover, according to tree distance error, which incorporates error at both the section and the paragraph level, the performance of the

HIERARCHICAL method is clearly superior. This result confirms the benefit of our selective update mechanism as well as the overall importance of joint learning.

Viewing human performance as an upper bound for machine performance highlights the gains of our algorithm. We observe that the gap between our method and human performance at the paragraph level is 32% smaller than that between the PIPELINE model and human performance, as well as the FLAT model and human performance.

Sentence-level Evaluation Until this point, we have evaluated the accuracy of insertions at the paragraph level, remaining agnostic as to the specific placement within the predicted paragraph. We perform one final evaluation to test whether the global hierarchical view of our algorithm helps in determining the *exact* insertion point. To make sentence-level insertion decisions, we use a local model in line with previous sentence-ordering work (Lapata, 2003; Bollegala et al., 2006). This model examines the two surrounding sentences of each possible insertion point and extracts a feature vector that includes lexical, positional, and temporal properties. The model weights are trained using the standard ranking perceptron (Collins, 2002).

We apply this local insertion model in two different scenarios. In the first, we ignore the global hierarchical structure of the document and apply the local insertion model to every possible sentence pair. Using this strategy, we recover 24% of correct insertion points. The second strategy takes advantage of global document structure by first applying our hierarchical paragraph selection method and only then applying the local insertion to pairs of sentences within the selected paragraph. This approach yields 35% of the correct insertion points. This statistically significant difference in performance indicates that purely local methods are insufficient when applied to complete real-world documents.

7 Conclusion and Future Work

We have introduced the problem of sentence insertion and presented a novel corpus-based method for this task. The main contribution of our work is the incorporation of a rich hierarchical text representation into a flexible learning approach for text structuring. Our learning approach makes key use of the hierarchy by selecting to update only the layer found responsible for the incorrect prediction. Empirical tests on a large collection of real-world insertion data confirm the advantage of this approach.

Sentence ordering algorithms too are likely to benefit from a hierarchical representation of text. However, accounting for long-range discourse dependencies in the unconstrained ordering framework is challenging since these dependencies only appear when a particular ordering (or partial ordering) is considered. An appealing future direction lies in simultaneously inducing hierarchical and linear structure on the input sentences. In such a model, tree structure could be a hidden variable that is influenced by the observed linear order.

We are also interested in further developing our system for automatic update of Wikipedia pages. Currently, our system is trained on insertions in which the sentences of the original text are not modified. However, in some cases additional text revisions are required to guarantee coherence of the generated text. Further research is required to automatically identify and handle such complex insertions.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865) and the Microsoft Research Faculty Fellowship. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF. Thanks to S.R.K. Branavan, Eugene Charniak, Michael Collins, Micha Elsner, Jacob Eisenstein, Dina Katabi, Igor Malioutov, Christina Sauper, Luke Zettlemoyer, and the anonymous reviewers for helpful comments and suggestions. Data used in this work was collected and processed by Christina Sauper.

References

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the ACL*, pages 141–148.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of HLT-NAACL*, pages 113–120.

Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *JAIR*, 17:35–55.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2006. A bottom-up approach to sentence ordering for multi-document summarization. In *Proceedings of the COLING/ACL*, pages 385–392.

Lijuan Cai and Thomas Hofmann. 2004. Hierarchical document categorization with support vector machines. In *Proceedings of the CIKM*, pages 78–87.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. 2006a. Hierarchical classification: Combining bayes with SVM. In *Proceedings of the ICML*, pages 177–184.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. 2006b. Incremental algorithms for hierarchical classification. *JMLR*, 7:31–54.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the ACL*, pages 111–118.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP*, pages 1–8.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

Ofer Dekel, Joseph Keshet, and Yoram Singer. 2004. Large margin hierarchical classification. In *Proceedings of the ICML*, pages 209–216.

Pablo Duboue and Kathleen McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the EMNLP*, pages 121–128.

Micha Elsner and Eugene Charniak. 2007. A generative discourse-new model for text coherence. Technical Report CS-07-04, Brown University.

Micha Elsner, Joseph Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *Proceedings of the HLT-NAACL*, pages 436–443.

Marti Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the ACL*, pages 9–16.

Nikiforos Karamanis, Massimo Poesio, Chris Mellish, and Jon Oberlander. 2004. Evaluating centering-based metrics of coherence for text structuring using a reliably annotated corpus. In *Proceedings of the ACL*, pages 391–398.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the ACL*, pages 545–552.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, LREC*, pages 48–56.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the ACL*, pages 69–76.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the ACL*, pages 368–375.

Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. 2004. Improving chronological sentence ordering by precedence relation. In *Proceedings of the COLING*, pages 750–756.

Ehud Reiter and Robert Dale. 1990. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the HLT-NAACL*, pages 149–156.

Automatically Identifying the Arguments of Discourse Connectives

Ben Wellner[†*]
[*]The MITRE Corporation
202 Burlington Road
Bedford, MA USA

James Pustejovsky[†]
[†]Department of Computer Science
Brandeis University
Waltham, MA USA

Abstract

In this paper we consider the problem of automatically identifying the arguments of discourse connectives (e.g., *and*, *because*, *nevertheless*) in the Penn Discourse TreeBank(PDTB). Rather than identifying the full *extents* of these arguments as annotated in the PDTB, however, we re-cast the problem to that of identifying the argument *heads*, effectively side-stepping the problem of *discourse segmentation*. We demonstrate significant gains using features derived from a dependency parse representation over those derived from a constituent-based tree parse. By also capturing inter-argument dependencies using a log-linear re-ranking model we identify both arguments correctly for over 74% of the connectives on held-out test data using gold-standard parses.

1 Introduction

The study of discourse is concerned with analyzing how phrase, clause or sentence-level units of text are *related* to each other within a larger unit of text (e.g., a document). Long recognized as important in dialog and text generation, this level of analysis is important generally for applications needing to place events and propositions in their proper context such as scenario-level information extraction, question answering, summarization, sentiment analysis and others.

In line with much of the NLP research agenda, recently a number of annotated corpora have emerged which encode discourse-level phenomena, making it possible to apply supervised, empirically-driven techniques to identifying discourse relations. Such corpora include the RST Discourse Treebank (Carlson et al., 2003) (based on Rhetorical Structure Theory), the Discourse GraphBank (Wolf and Gibson, 2005) (based on the relations of Hobbs (1985)) and the Penn Discourse Treebank (Miltsakaki et al., 2004b). While these corpora differ in many ways, they all more or less encode problems involving: 1) identifying/segmenting the basic units of discourse (e.g., clauses, phrases), 2) determining for which pairs of segments (or segment groups) a discourse relation exists, and 3) characterizing the *type* of relation (cause, elaboration, etc.) between segment pairs.

For our experiments in this paper, we use the Penn Discourse TreeBank (PDTB). The PDTB differs from most other discourse-level annotation efforts in its bottom-up, lexically-driven approach. Rather than identifying all possible discourse relations, the PDTB focuses on annotating relations lexicalized by discourse connectives that explicitly occur in the text along with their two arguments. [1] These discourse connectives include coordinating conjunctions (e.g., *and*, *or*), subordinating conjunctions (e.g., *because*, *when*, *since*) and discourse adverbials (e.g., *however*, *previously*, *nevertheless*).

In this paper we focus on problems (1) and (2)

[1] The final release of the PDTB, scheduled for release in August 2007, will annotate the *type* of the rhetorical relation holding between arguments of explicit connectives in addition to annotating relations between adjacent sentences where no lexical connective is present.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 92–101, Prague, June 2007. ©2007 Association for Computational Linguistics

above. However, rather than explicitly identifying the discourse segments and then deciding for which pairs a relation exists, we focus on identifying relations between the pairs of *head words* that *represent* the discourse segments. In this sense, the problem resembles that of predicate-argument identification where the predicates are discourse connectives and the arguments are single words which serve as anchors for the discourse segments.

To address the problem of identifying the arguments of discourse connectives we incorporate a variety of lexical and syntactic features in a discriminative log-linear ranking model. To capture dependencies between the two arguments of a connective we use a log-linear *re*-ranking model to select the best argument pair from a set of N-best argument pairs provided by the independent argument models. Further, we provide an analysis of the contribution of the various features demonstrating that features based on a dependency parse representation outperform features derived from a constituent tree parse.

2 Overview of the Penn Discourse Treebank

Discourse arguments in the PDTB represent abstract objects (Asher, 1993) which include facts, propositions and events. Each argument must include at least one predicate and can be realized as: a clause, a VP within VP coordination, a nominalization (in certain, restricted cases), an anaphoric expression or a response to a question. Each connective has two arguments: Arg2 is the argument syntactically connected to the connective in the same sentence and Arg1 is the other argument which may lie in the same sentence as the connective or, generally, anywhere prior in the discourse.

The PDTB contains a total of 18505 explicit connectives annotated with discourse arguments. The annotations are layered on top of the Penn TreeBank-II (PTB) parse trees and cover all 25 Wall Street Journal (WSJ) sections.

2.1 Examples

Below are a few examples from the PDTB. Each Arg1 is denoted in *italics* and each Arg2 is denoted in **bold**. The head-words for each argument are underlined. We discuss and motivate the identification of head-words in Section 2.2.

(1) *Choose 203 business executives, including, perhaps, someone from your own staff,* and **put them out on the streets**, to be deprived for one month of their homes, families and income.

(1) shows an example of a coordinating connective *and* and its two arguments. In this case, the Arg1 lies in the same sentence as the connective. It is also possible for the Arg1 to lie outside the sentence (usually in the immediately preceding sentence) when the coordinating connective begins a sentence.

An example of the subordinating connective, *because* is shown below in (2). This example brings up some interesting ambiguities that arise quite regularly in the data. An alternative reading for this example might only include the extent *to duck liability* for the Arg1. That is, the predicate *be able* could be read to include the discourse relation and its two arguments as an argument.

(2) *Drug makers shouldn't be able to duck liability* because **people couldn't identify precisely which identical drug was used.**

Both coordinating and subordinating connectives are *structural* (Webber et al., 2003). Discourse adverbials however, take one argument, Arg2, structurally but the other can be anaphoric: its Arg1 may be present anywhere in the current running discourse with little or no restriction. Example (3) shows the case in which the Arg1 lies in the previous sentence. In many cases, however, it resides in the same sentence as the connective or many sentences prior in the discourse.

(3) *France's second-largest government-owned insurance company, Assurances Generales de France, has been building its own Nagivation Mixte stake*, currently thought to be between 8% and 10%. Analysts said

they don't think it is contemplating a takeover, however, and its officials couldn't be reached.

2.2 Head-Based Representation of the PDTB

In contrast to other annotations layered on the PTB such as PropBank and NomBank, the arguments of a discourse connective generally do not correspond to a single parse tree constituent. Arguments consist instead of a *set* of non-overlapping constituents from the parse tree (i.e. a forest). This target representation makes the process of identifying the arguments to discourse connectives difficult since the space of candidate arguments extents is considerably larger than for PropBank parsing, for example. Even without this added difficulty, discourse segmentation is one of the most difficult stages in discourse parsing (Soricut and Marcu, 2003). While the segments themselves may be useful in certain contexts, for many applications, if not most, it will still be necessary to *interpret* these segments (e.g. at the predicate-argument level). As such, we argue that, in general, identifying the lexical *heads* of these discourse segments is sufficient and perhaps even preferable for this stage of processing. A problem arises, however, with arguments that consist of sequences of abstract objects represented as coordinated or subordinated sequences of VPs, clauses or sentences. What should the head be in such cases? By convention we designate the extent head as the head of the first element in the sequence. In (4), the head of the Arg2 would be *went*, but it's implicit scope includes the second VP coordinate headed by *caught*.

(4) Mr. Dozen even related *the indignity suffered* when **he and two colleagues went on an overnight fishing expedition of the New Jersey shore and caught nothing.**

The problem then becomes how to determine the end of the sequence of abstract objects. In many cases, there is a "natural end" to such sequences based on the syntax. In (4), the natural end is simply the end of the VP coordination. Difficult cases remain, however, particularly with multi-sentential Arg1s of anaphoric connectives. Determining the end of the these arguments seems non-trivial.[2] Nevertheless, identifying the begininning of the argument (via its head) is an important step in modeling these difficult cases.

2.3 Head Identification

Identifying the head of a discourse argument given its extent (as described by a set of constituent sub-trees in the PTB) consists of two steps. First, we construct a single syntactic tree formed by taking all of the sub-trees in the extent, finding their least common ancestor (LCA) node and including all intermediate nodes from the subtrees to the LCA node. Then, a slight variation of the head finding algorithm in (Collins, 1999) is applied to the derived tree to find the head. Figure 1 provides an example indicating the arguments to the connective "After" and the derived argument heads.

3 Discourse Argument Identification

Identifying the arguments of discourse connectives can be naturally formulated as a binary classification task where separate classifiers are trained for each argument — i.e., Arg1 and Arg2. First, a set of candidate arguments, α_i is gathered for each connective, π. Training instances, $\langle \alpha_i, \pi \rangle$, are then created for each candidate with respect to the connective. A training instance is positive if α_i is the true argument for π and negative otherwise. At decoding time, the candidate classified positively with the highest probability (or score) compared to the other candidates is selected as the argument.

An alternative to using a standard classification approach is to use a *ranking* model. The advantage of the ranking model is that candidate instances are compared against each other *during training* as well as during decoding. In

[2] There are indications, however, that the end of the argument sometimes falls out of the (possibly non-lexicalized) discourse relations local to the argument.

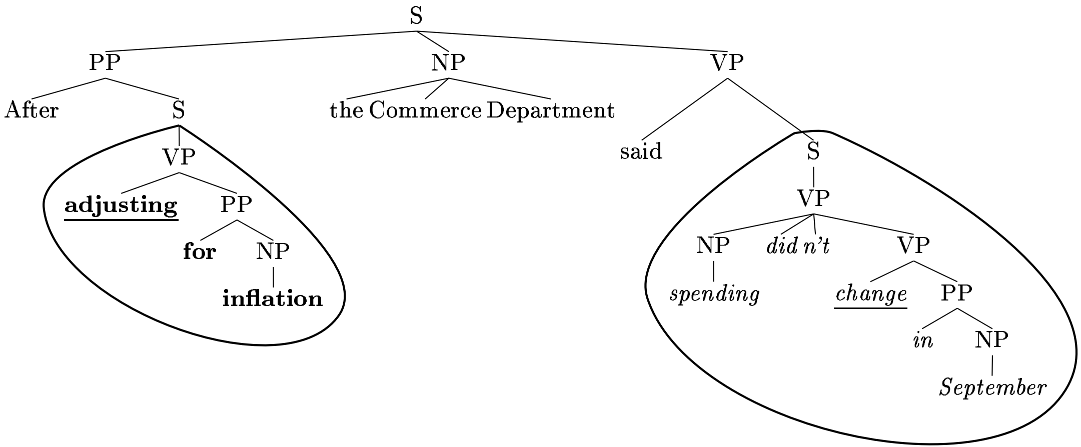


Figure 1: Syntactic structure and discourse arguments for the connective "After".

contrast, with a standard classifier, separate instances (i.e. candidates) are trained and classified as if they were completely independent. We use a log-linear ranking model. Such models have been used for a variety of other tasks including co-reference (Denis and Baldridge, 2007), question answering (Ravichandran et al., 2003) and parse re-ranking (Charniak and Johnson, 2005). For a given ARG1 candidate, α_i, the probability of that candidate being the argument given the connective, π, and the document, x, is defined according to the model as:

$$P_1(\alpha_i|\pi,x) = \frac{\exp\left(\sum_k \lambda_k f_k(\alpha_i,\pi,x)\right)}{\displaystyle\sum_{\alpha_j \in C_1(\pi,x)} \exp\left(\sum_k \lambda_k f_k(\alpha_j,\pi,x)\right)} \qquad (1)$$

where the f_k are feature functions, the λ_k are their weights and $C_1(\pi,x)$ is the set of candidate ARG1 arguments for the connective π in the document x. The model for ARG2 is defined analogously, but may in fact use a different set of features or a different candidate generation function. At training time, all potential candidates of a particular type for a given connective are provided to the ranking model as a distribution: the correct gold-standard candidate receiving a probability mass of 1.0 and the other candidates receiving masses of 0.0. During decoding, we select candidates in the same way as for training and produce a distribution over these candidates according to equation 1, selecting the candidate assigned the highest probability by the model as the argument.

We compared both the above ranking model and a standard binary Maximum Entropy model (i.e., logistic regression) and found the ranking model to have a small but consistent edge over the classifier. Accordingly, we only report results here using the ranking model.

3.1 Candidate Selection

Selecting the candidate arguments, α_i, is an important aspect of the problem. There are conceivably very many possible ARG1 candidates for a given connective stretching back from the sentence containing the connective to the beginning of the document. We employ two simple criteria to reduce the space of candidate argument head words. First, we only consider argument candidates that have an appropriate part-of-speech (all verbs, common nouns, adjectives). Second, we only consider candidates that are within 10 "steps" of the connective where a single step includes a sentence boundary or a syntactic dependency link within a sentence (see Figure 2). Only candidates lying within the same sentence as the connective are considered for ARG2.

3.2 Features

We used a variety of features for identifying the discourse arguments of a connective.

Baseline Features. Our baseline features included simply the connective and argument words, where the connective appears in the sentence, whether the argument precedes or follows the connective and whether the argument is in the same sentence as the connective or not.

Constituent Path Features. As noted in work on semantic role labeling, features derived from the constituent parse of the sentence can be very helpful for deriving the argument structure of predicating verbs (Toutanova et al., 2005) and nouns (Jiang and Ng, 2006). Syntax plays a strong role in identifying discourse arguments, too, though even for structural connectives it by no means "aligns" with the discourse structure (Dinesh et al., 2005). We introduced a feature capturing the constituent tree path from the connective to the candidate argument as well as variants in which repeated nodes and part-of-speech nodes are removed from the path. If the argument lies in a different sentence, the path from the connective to the argument consists of the path from the connective to the top node of its sentence, followed by a series of virtual *SENT* nodes for the intervening sentences and then ending with the path from the top node of the sentence containing the argument to the argument head itself.

Dependency Path Features. We experimented with a number of syntactic features based on a *dependency* parse representation. The primary motivation here being that it provides for a more compact and natural representation of the syntax, providing for better syntactic features with less data sparseness than constituent path features. The dependency representation we use is that put forth in de Marneffe et al. (2006) and we apply their approach to deriving the dependency structure from the constituent parse. The features used here include the (shortest) dependency path from the connective to the prospective argument and two collapsed versions removing coordination links as well as repeated links of the same type. For argument candidates in prior sentences, we introduce *SENT* links for each intervening sentence.

Connective Features. Different discourse connectives behave differently depending on their type. A potentially important feature then involves capturing the connective type: (coordinating, subordinating or adverbial). We use the categorized lists of discourse connectives found in (Knott, 1996); further, any connectives not appearing in these lists are considered discourse adverbials. As we would expect different syntax associated with different connectives we introduce conjunctive features such as the connective type and syntactic path.

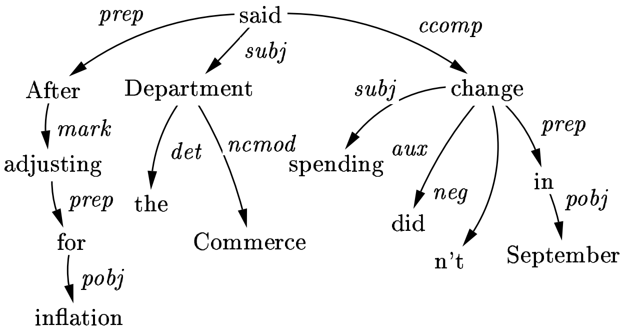


Figure 2: Dependency structure.

Lexico-Syntactic Features. One of the prime difficulties in identifying the correct non-anaphoric argument has to do with *attribution.* In this situation the argument is the complement of a verb indicating attribution of the proposition denoted by the complement to an individual other than the writer. Figure 1 provides an example of this where the ARG1 of "After" is the complement of the verb "said" being attributed to "the Commerce Department". To model this situation we introduce features capturing whether the argument is a potentially attribution-denoting verb, whether it has a clausal complement, whether it is the clausal complement of another verb and whether the complementing verb is attributing.

A full listing of the features used for identifying arguments is shown in Table 1.

4 Experiments with Independent Argument Identification

For all of our experiments, we use sections 02-22 for training, sections 00-01 for development and sections 23-24 for testing. The development data was used to customize our features and to tune the Gaussian prior used to prevent over-

Baseline Features	
A	Where in the sentence (beginning, middle, end) the connective resides
B	Whether the argument is in the same sentence as the connective (yes,no)
C	Connective phrase
D	Downcase connective phrase
E	Argument head word
F	Argument head prior or after connective
G	A & B
Constitutent Features	
H	Path from argument to connective through the constituent tree
I	Length of path
J	Collapsed path without part-of-speech
K	Collapsed path removing repetitions of the same node type (e.g. VP-VP-VP → VP)
L	C & H
Dependency Features	
M	Dependency path from argument to connective
N	Path + head word of first link from connective
O	Collapsed path removing coordinating links
P	Collapsed path removing repetitions of links
Q	C & M
Connective Features	
R	coordinating, subordinating or adverbial connective
S	A & R
T	M & R
Lexico-Syntactic Features	
U	Argument is an attributing verb
V	Argument has a clausal complement
W	U & V
X	Argument is a clausal complement of a verb
Y	X & governing verb is an attributing verb

Table 1: Feature types for discourse connective argument identification

fitting in the log-linear models (at $\sigma = 0.25$ for both the local and the re-ranking models). All results are reported on the testing data, sections 23-24. We report results using both gold-standard parses and automatic parses using the Charniak-Johnson parser (Charniak and Johnson, 2005).

For evaluating Arg1 and Arg2 argument identification performance we report *accuracy* — i.e., the percentage of arguments correctly identified. An argument is correct if and only if it is the same head-word as derived from the argument extent as annotated in the PDTB (as described in Section 2.3). We also report *Connective Accuracy* which is the percentage of connectives for which *both* arguments were correctly identified.

	Accuracy		
FeatureSet	Arg1	Arg2	Conn.
A-G	32.7	60.7	21.6
A-L	60.6	85.5	53.6
A-G;M-Q	73.7	94.2	70.2
A-Y	75.0	94.2	71.7
A-Y(auto)	67.9	90.6	62.7

Table 2: Results for argument identification on the testing data (WSJ sections 23-24) gold standard parses (with various feature sets) and Charniak-Johnson parses (auto) for the full feature set A-Y.

Our results for the task of identifying arguments are shown in Table 4 for various feature combinations. It is interesting to compare the performance of the constituent parse features (A-L) vs. the dependency parse features (A-G;M-Q). The dependency parse features perform markedly better: 70.2 vs. 53.6 Connective Accuracy with gold-standard parses.

5 Experiments With Re-ranking

A drawback to the above approach is that the two arguments are identified independently. Ideally, one would like to consider both arguments and the connective simultaneously, taking into account global properties such as the pattern of the argument structure (e.g. Connective-Arg2-Arg1 vs. Arg1 Connective Arg2) or properties of compatibility between the two arguments (e.g. agreement in tense). Considering all pairs of arguments outright, however, presents scalability issues as the number of such pairs can be very large (especially with anaphoric Arg1s). Indeed, a huge advantage of the lexicalized approach taken with the PDTB is that we *can* identify arguments independently using the connectives as anchors. Nevertheless, there is obvious potential gain from modeling pairs of arguments jointly.

One way to model these dependencies in a tractable fashion is to use a *re-ranking* approache (Collins, 2000) which has proven successful in a variety of NLP tasks. The basic idea is to use a model with strong independenc as-

	Accuracy		
N	ARG1	ARG2	Conn.
1	74.5	94.5	71.4
5	83.1	97.4	81.8
10	90.5	97.9	89.2
20	93.8	97.9	92.1
30	94.6	97.9	92.9

Table 3: N-best upper-bounds for different values of N according to a product of independent argument ranker probabilities with the full feature set (A-Y)

sumption, $GEN(\pi)$, in this case based on the independent argument models described above, to generate N candidate argument pairs for a given connective, π. Then, the re-ranking model is used to re-rank these candidate pairs; the top-ranked pair is then selected.

In our setting for a given connective, π, we define the *local probability* for a candidate argument pair, $\langle\alpha_i, \alpha_j\rangle$ as:

$$P_{loc}(\alpha_i, \alpha_j|\pi, x) = P_{\text{ARG1}}(\alpha_i|\pi, x)\cdot P_{\text{ARG2}}(\alpha_j|\pi, x)$$

Thus, $GEN(\pi)$ generates the top N argument pairs according to the P_{loc}. In practice, we also assert that $P_{loc}(\alpha_j, \alpha_k|\pi, x) = 0$ when $j = k$.

For different values of N, Table 3 shows the oracle upper bounds on performance - the performance achieved by selecting the correct argument pair from $GEN(\pi)$ if it is in the list of argument pairs and otherwise selecting the first pair with one correct argument if such a pair exists. Note that performance on ARG2 plateaus at 97.9. This is due to 2.1 percent of the ARG2s not being reachable because they are not considered candidates (they are more than 10 “parse steps” away or an invalid part-of-speech).

5.1 Modeling Inter-Argument Dependencies

The model for re-ranking pairs of arguments is given by

$$P_r(\alpha_i, \alpha_j|\pi, x) = \frac{\exp\left(\sum_k \lambda_k f_k(\alpha_i, \alpha_j, \pi, x)\right)}{\sum_{\alpha_i, \alpha_j \in GEN(\pi)} \exp\left(\sum_k \lambda_k f_k(\alpha_i, \alpha_j, \pi, x)\right)}$$

Following previous work (Collins, 2000; Toutanova et al., 2005), we mix the local model into the final score along with the re-ranking model as:

$$P(\alpha_i, \alpha_j|\pi, x) = P_{loc}(\alpha_i, \alpha_j|\pi, x)^{\gamma}\cdot P_r(\alpha_i, \alpha_j|\pi, x)$$

where γ indicates the degree to which the local model influences the final score. Tuning γ on the development data, we set $\gamma = 0.4$ for all our re-ranking experiments.

The re-ranking model is able to accommodate features over *both* candidate arguments. For example, we can test whether the two arguments are the same predicate or whether they are both reporting verbs. Another set of features consists of triples denoting the relative order of the arguments and the connective. For example, the feature *CONN*_ARG2_ARG1 indicates the connective and both arguments lie in the same sentence with the connective first, followed by ARG2 and then ARG1. The feature *Prev_CONN*_ARG2 indicates ARG1 is in the previous sentence and the connective precedes ARG2 within the sentence containing the connective. Other slight variations capture configurations where the ARG1 candidate lies further back in the discourse. Finally, we found some utility in comparing the syntactic arguments (e.g., subject, direct object) of the candidate argument pairs. For example, the arguments of the discourse adverbial *also* not only frequently involve the same predicate but also involve the same entities that appear as arguments to the predicate. Currently, we simply introduce features testing whether the argument strings are identitical as a proxy for full co-reference.

Table 4 shows the results incorporating the re-ranking model for the different feature sets described earlier. The re-ranking models in each case are constructed from the features that would naturally be available to the re-ranker. For example, the re-ranking model for feature set A-Y uses a feature testing whether both candidate arguments are reporting verbs, whereas the re-ranking model for A-L doesn’t.

	Accuracy			
Features	Arg1	Arg2	Conn.	Err.
A-G	44.1	59.6	30.6	11.5%
A-L	64.7	85.6	58.1	9.6%
A-G;M-Q	74.2	94.4	71.8	5.4%
A-Y	76.4	95.4	74.2	8.8%
A-Y(auto)	69.8	90.8	64.6	5.4%

Table 4: Re-ranking results for argument identification on the testing data using gold-standard and Charniak-Johnson parses for the full feature set, A-Y (auto). The error reduction (Err.) is relative to the results in Table 2.

5.2 Discussion and Error Analysis

Not surprisingly, performance at identifying Arg2s is much higher than for Arg1s as the former are syntactically bound to the connective. Indeed, performance for identifying Arg2s may be at or very close to human levels of performance using gold-standard parses. Miltsakaki et al. (2004a) indicate 94.1% inter-annotator agreement for Arg2, 86.3% on Arg1 and 82.8% agreement per discourse connective with respect to the full argument extents for a set of 10 connectives. The disagreement rates, however, would likely be reduced considerably using our head-based representation since almost half of the disagreements reported were due to argument extent disagreements.

Many of the Arg2 errors we found had to do with attribution, such as:

(5) ..“We pretty much *have* a policy of not commenting on rumors, and I **think(?)** that **falls** in that category.

where the system proposed “think” as the Arg2 and the annotated argument was “falls”.

The Arg1 errors were much more diverse with many involving arguments in previous sentences, such as the following case in which the system proposed *owned* as the argument yet the correct argument was *completed* found three sentences prior in the discourse.

(6) ..Quantum *completed* in August an acquisition of Petrolane... Petrolane is the second-largest... The largest, Suburban Propane, was already *owned(?)* by Quantum. Still, Quantum **has** a crisis to get past right now.

Conn. Type	Freq.	Indep. Acc.	Rerank Acc.	Err.
Coord.	662	75.5	78.3	11.4%
Subord.	547	87.2	86.8	-3.0%
Adv.	386	42.2	49.0	11.8%
Total	1595	71.7	74.2	8.8%

Table 5: Frequency of each connective type and connective accuracy for the independent and re-ranking approaches using gold-standard parses and features (A-Y).

An examination of the errors by connective type is shown in Table 5. The re-ranking model provides considerable improvement for coordinating and adverbial connectives, but slighly *lowers* performance for subordinating connectives. Overall performance on discourse adverbials remains below 50% however.

6 Related Work

Given the formulation of discourse relations as predicate-argument structures anchored on discourse connectives, our work here bears some resemblance to work in semantic role labeling that has focused on identifying semantic frames for verbs (Toutanova et al., 2005). The task of identifying discourse relations is simpler in that there are only and exactly two arguments for each predicate; yet it is more difficult due to many more candidate arguments not contained within a single sentence.

Within discourse parsing, our work is similar to that of Soricut and Marcu (2003) but they focus only on identifying (and labeling the type of) all intra-sentential discourse relations whereas we attempt to identify discourse relations spanning multiple sentences, provided they are lexicalized by a connective. While not directly comparable to our results, they report 73.0 F-measure at identifying intra-sentential discourse relations and segments using gold-standard parses. With gold-standard discourse segments provided, their system achieves human-levels of performance (96.2

F-measure), broadly comparable to our near-human levels of performance on identifying Arg2s with gold-standard parses. Sporleder and Lapata (2005) address intra-sentential discourse modeling with a chunking approach. They achieve 88.7 F-measure on identifying discourse segment boundaries and 76.3 F-measure when also labeling each segment as a nucleus or satellite. Webber et al. (2003) provide a discourse parsing model, DLTAG, which is an extension of Lexicalized Tree Adjoining Grammars. Baldridge and Lascarides (2005) present a discourse parser for dialogue in the framework of SDRT (Asher, 1993) and achieve 67.9 F-measure on identifying and segmenting discourse relations.

7 Conclusions and Future Work

We have presented a fully automated system capable of identifying the arguments of discourse connectives. Rather than identifying the full argument extents in the PDTB, we have proposed here an alternative problem formulation: that of identifying the heads of discourse arguments. [3] With such a representation our system achieves 74.2% accuracy using gold-standard parses and 64.6% accuracy using automatic parses on the task of correctly identifying both arguments of discourse connectives. We found that syntactic features based on a dependency parse representation provide more discriminative features over those based on a constituent tree representation. Additionally, we found a notable improvement by exploiting joint features over argument pairs in a re-ranking model in comparison to modeling the arguments independently.

We have provided here, to our knowledge, the first rigorous empirical results on identifying the arguments of discourse connectives in the PDTB. Accordingly, many avenues remain for future work. Further feature engineering, particularly work capturing the lexico-semantic, attributive and predicate-argument properites of arguments appears necessary to better identify the Arg1s of anaphoric discourse adverbials, in particular. Introducing separate models and feature sets for each of the three connective types may also prove beneficial since phenomena involved vary according to connective type.

While we have demonstrated some encouraging results by modeling both arguments jointly, we hypothesize more gains are possible by modeling *inter-connective* dependencies. The discourse arguments of one connective are not independent of other (nearby) connectives and their arguments. For example, it is very rare to see crossing argument links. Capturing these inter-connective dependencies and constraints is likely to be even more important when considering the task of identifying the rhetorical *types* associated with the connectives or when considering non-lexicalized relations between adjacent sentences.

Finally, jointly modeling PropBank and the PDTB is another interesting area we plan to investigate, something to which the head-based approach and dependency parse representation we advocate here would be well-suited.

Acknolwedgements

The first author was supported in part by MITRE Sponsored Research. The authors thank Bonnie Webber, Aravind Joshi, Alan Lee and Inderjeet Mani for useful discussions and commments on an earlier draft of this paper.

References

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning CoNLL-2005*.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current Directions in Discourse and Dialogue*, pages 85–112. Kluwer Academic Publishers.

[3] Software for producing the head-based representation of the PDTB, an augmented version of the Charniak-Johnson parser that a produces dependency representation, and the log-linear ranking code are available at: http://www.cs.brandeis.edu/w̃ellner/pdtb-emnlp/

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics.*

M. Collins. 1999. Head-driven statistical models for natural language parsing.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML-2000.*

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation*, Genoa, Italy.

Pascal Denis and Jason Baldridge. 2007. A ranking approach to pronoun resolution. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007).*

Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Attribution and the (non)-alignment of syntactic and discourse arguments of connectives. In *Proceedings of the ACL Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, Ann Arbor, Michigan, USA.

Jerry Hobbs. 1985. On the coherence and structure of discourse.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic role labeling of nombank: A maximum entropy approach. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, Sydney, Australia.

Alistair Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations.* Ph.D. thesis, University of Edinburgh.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004a. Annotating discourse connectives and their arguments. In *Proceedings of the HLT/NAACL Workshop on Frontiers in Corpus Annotation.*

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004b. The penn discourse treebank. In *LREC 2004.*

Deepak Ravichandran, Eduard Hovy, and Franz Josef Och. 2003. Statistical qa - classifier vs. re-ranker: What's the difference? In *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering-Machine Learning and Beyond.*

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, Edmonton, Canada.

Caroline Sporleder and Mirella Lapata. 2005. Discourse chunking and its application to sentence compression. In *Proceedings of the HLT/EMNLP*, pages 257–264, Vancouver.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the Association for Computational Linguistics(ACL)*, Ann Arbor, Michigan, USA.

Bonnie Webber, Aravind Joshi, Alistair Knott, and Matthew Stone. 2003. Anaphora and discourse structure. *Computational Linguistics.*

F. Wolf and E. Gibson. 2005. Representing discourse coherence: A corpus-based analysis. *Computational Linguistics*, 31(2):249–287.

Incremental generation of plural descriptions: Similarity and partitioning

Albert Gatt and **Kees van Deemter**
Department of Computing Science
University of Aberdeen
{agatt,kvdeemte}@csd.abdn.ac.uk

Abstract

Approaches to plural reference generation emphasise descriptive brevity, but often lack empirical backing. This paper describes a corpus-based study of plural descriptions, and proposes a psycholinguistically-motivated algorithm for plural reference generation. The descriptive strategy is based on partitioning and incorporates corpus-derived heuristics. An exhaustive evaluation shows that the output closely matches human data.

1 Introduction

Generation of Referring Expressions (GRE) is a well-studied sub-task of microplanning in Natural Language Generation. Most algorithms in this area view GRE as a content determination problem, that is, their emphasis is on the construction of a semantic representation which is eventually mapped to a linguistic realisation (i.e. a noun phrase). Content Determination for GRE starts from a Knowledge Base (KB) consisting of a set of entities U and a set of properties $\mathbb{P}$ represented as attribute-value pairs, and searches for a description $D \subseteq \mathbb{P}$ which distinguishes a referent $r \in U$ from its distractors. Under this view, reference is mainly about *identification* of an entitiy in a given context (represented by the KB), a well-studied pragmatic function of definite noun phrases in both the psycholinguistic and the computational literature (Olson, 1970).

For example, the KB in Table 1 represents 8 entities in a $2D$ visual domain, each with 6 attributes, including their location, represented as a combination of horizontal (X) and vertical (Y) numerical coordinates. To refer to an entity an algorithm searches through values of the different attributes.

	TYPE	COLOUR	ORIENTATION	SIZE	X	Y
e_1	desk	red	back	small	3	1
e_2	sofa	blue	back	small	5	2
e_3	desk	red	back	large	1	1
e_4	desk	red	front	large	2	3
e_5	desk	blue	right	large	2	4
e_6	sofa	red	back	large	4	1
e_7	sofa	red	front	large	3	3
e_8	sofa	blue	back	large	3	2

Table 1: A visual domain

GRE has been dominated by Dale and Reiter's (1995) Incremental Algorithm (IA), one version of which, generalised to deal with *non-disjunctive* plural references, is shown in Algorithm 1 (van Deemter, 2002). A non-disjunctive reference to a set R is possible just in case all the elements of R can be distinguished using the same attribute-value pairs. Such a description is equivalent to the logical conjunction of the properties in question. This algorithm, IA$_{plur}$, initialises a description D and a set of distractors C [1.1–1.2], and traverses an ordered list of properties, called the *preference order* ($\mathcal{PO}$) [1.3], which reflects general or domain-specific pref-

Algorithm 1 IA$_{plur}$(R,U,$\mathcal{PO}$

```
1:  $D \leftarrow \emptyset$
2:  $C \leftarrow U - R$
3:  for $\langle A : v \rangle \in \mathcal{PO}$ do
4:      if $R \subseteq \llbracket \langle A : v \rangle \rrbracket \wedge \llbracket \langle A : v \rangle \rrbracket - C \neq \emptyset$ then
5:          $D \leftarrow D \cup \{\langle A : v \rangle\}$
6:          $C \leftarrow C \cap \llbracket \langle A : v \rangle \rrbracket$
7:          if $\llbracket D \rrbracket = R$ then
8:              return $D$
9:          end if
10:     end if
11: end for
12: return $D$
```

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 102–111, Prague, June 2007. ©2007 Association for Computational Linguistics

erences for attributes. For instance, with the $\mathcal{PO}$ in the top row of the Table, the algorithm first considers values of TYPE, then COLOUR, and so on, adding a property to D if it is true of the intended referents R, and has some *contrastive value*, that is, excludes some distractors [1.4]. The description and the distractor set C are updated accordingly [1.5–1.6], and the description returned if it is distinguishing [1.7]. Given $R = \{e_1, e_2\}$, this algorithm would return the following description:

(1) $\langle$ORIENTATION : *back*$\rangle \wedge \langle$SIZE : *small*$\rangle$

This description is overspecified, because ORIENTATION is not strictly necessary to distinguish the referents ($\langle$SIZE : *small*$\rangle$ suffices). Moreover, the description does not include TYPE, though it has been argued that this is always required, as it maps to the head noun of an NP (Dale and Reiter, 1995). We will adopt this assumption here, for reasons explained below. Due to its hillclimbing nature, the IA avoids combinatorial search, unlike some predecessors which searched exhaustively for the briefest possible description of a referent (Dale, 1989), based on a strict interpretation of the Gricean Maxim of Quantity (Grice, 1975). Given that, under the view proposed by Olson (1970) among others, the function of a referential NP is to identify, a strict Gricean interpretation holds that it should contain no more information than necessary to achieve this goal.

The Incremental Algorithm constitutes a departure from this view given that it can overspecify through its use of a $\mathcal{PO}$. This has been justified on psycholinguistic grounds. Speakers overspecify their descriptions because they begin their formulation of a reference without exhaustively scanning a domain (Pechmann, 1989; Belke and Meyer, 2002). They prioritise the basic-level category (TYPE) of an object, and salient, absolute properties like COLOUR (Pechmann, 1989; Eikmeyer and Ahlsèn, 1996), as well as locative properties in the vertical dimension (Arts, 2004). Relative attributes like SIZE are avoided unless absolutely required for identification (Belke and Meyer, 2002). This evidence suggests that speakers conceptualise referents as *gestalts* (Pechmann, 1989) whose core is their basic-level TYPE (Murphy, 2002) and some other salient attributes like COLOUR. For instance, according to Schriefers and Pechmann (1988), an NP such as *the large black triangle* reflects a conceptualisation of the referent as a *black triangle*, of which the SIZE property is predicated. Thus, the TYPE+COLOUR combination is not mentally represented as two separable dimensions.

In what follows, we will sometimes refer to this principle as the *Conceptual Gestalts Principle*. Note that the IA does not fully mirror these human tendencies, since it only includes preferred attributes in a description if they remove some distractors given the current state of the algorithm, whereas psycholinguistic research suggests that people include them irrespective of contrastiveness (but cf. van der Sluis and Krahmer (2005)).

More recent research on plural GRE has de-emphasised these issues, especially in case of *disjunctive* plural reference. Disjunction is required whenever elements of a set of referents R do not have identical distinguishing properties. For example, $\{e_1, e_3\}$ can be distinguished by the following Conjunctive Normal Form (CNF) description[1]:

(2) $\langle$TYPE : *desk*$\rangle \wedge \big(\langle$COLOUR : *red*$\rangle \vee \langle$COLOUR : *blue*$\rangle\big) \wedge \big(\langle$ORIENTATION : *right*$\rangle \vee \langle$ORIENTATION : *back*$\rangle\big)$

Such a description would be returned by a generalised version of Algorithm 1 proposed by van Deemter (2002). This generalisation, IA$_{bool}$ (so called because it handles all Boolean operators, such as negation and disjunction), first tries to find a non-disjunctive description using Algorithm 1. Failing this, it searches through disjunctions of properties of increasing length, conjoining them to the description. This procedure has three consequences:

1. **Efficiency**: Searching through disjunctive combinations results in a combinatorial explosion (van Deemter, 2002).

2. **Gestalts and content**: The notion of a 'preferred attribute' is obscured, since it is difficult to apply the same reasoning that motivated the $\mathcal{PO}$ in the IA to combinations like (COLOUR $\vee$ SIZE).

[1] Note that logical disjunction is usually rendered as linguistic coordination using *and*. Thus, *the table and the desk* is the union of things which are desks *or* tables.

3. **Form**: Descriptions can become logically very complex (Gardent, 2002; Horacek, 2004).

Proposals to deal with (3) include Gardent's (2002) non-incremental, constraint-based algorithm to generate the briefest available description of a set, an approach extended in Gardent et al. (2004). An alternative, by Horacek (2004), combines best-first search with optimisation to reduce logical complexity. Neither approach benefits from empirical grounding, and both leave open the question of whether previous psycholinguistic research on singular reference is applicable to plurals.

This paper reports a corpus-based analysis of plural descriptions elicited in well-defined domains, of which Table 1 is an example. This study falls within a recent trend in which empirical issues in GRE have begun to be tackled (Gupta and Stent, 2005; Jordan and Walker, 2005; Viethen and Dale, 2006). We then propose an efficient algorithm for the generation of references to arbitrary sets, which combines corpus-derived heuristics and a partitioning-based procedure, comparing this to IA$_{bool}$. Unlike van Deemter (2002), we only focus on disjunction, leaving negation aside. Our starting point is the assumption that plurals, like singulars, evince preferences for certain attributes as predicted by the Conceptual Gestalts Principle. Based on previous work in Gestalt perception (Wertheimer, 1938; Rock, 1983), we propose an extension of this to sets, whereby plural descriptions are preferred if (a) *they maximise the similarity of their referents*, using the same attributes to describe them as far as possible; (b) prioritise salient ('preferred') attributes which are central to the conceptual representation of an object. We address (3) above by investigating the logical form of plurals in the corpus. One determinant of logical form is the basic-level category of objects. For example, to refer to $\{e_1, e_2\}$ in the Table, an author has at least the following options:

(3) (a) the small desk and sofa
(b) the small red desk and the small blue sofa
(c) the small desk and the small blue sofa
(d) the small objects

These descriptions exemplify three possible sources of variation:

Disjunctive/Non-disjunctive: The last description, (3d), is non-disjunctive (i.e. it is logically a conjunction of properties). This, however, is only achievable through the use of a non-basic level value for the TYPE of the entities (*objects*). Using the basic-level would require the disjunction ($\langle$TYPE : *desk*$\rangle \vee \langle$TYPE : *sofa*$\rangle$), which is the case in (3a–c). Given that basic-level categories are preferred on independent grounds (Rosch et al., 1976), we would expect examples like (3d) to be relatively infrequent.

Aggregation: If a description is disjunctive, it may be aggregated, with properties common to all objects realised as wide-scope modifiers. For instance, in (3a), *small* modifies *desk and sofa*. By contrast, (3b) is *non-aggregated*: *small* occurs twice (modifying each coordinate in the NP). Non-aggregated, disjunctive descriptions are logically equivalent to a partition of a set. For instance, (3c) partitions the set $R = \{e_1, e_2\}$ into $\{\{e_1\}, \{e_2\}\}$, describing each element separately. Descriptions like (3b) are more overspecified than their aggregated counterparts due to the repetition of information.

Paralellism/Similarity: Non-aggregated, disjunctive descriptions (partitions) may exhibit *semantic parallelism*: In (3b), elements of the partition are described using exactly the same attributes (that is, TYPE, COLOUR, and SIZE). This is not the case in (3c), which does represent a partition but is *non-parallel*. Parallel structures maximise the similarity of elements of a partition, using the same attributes to describe both. The likelihood of propagation of an attribute across disjuncts is probably dependent on its degree of salience or preference (e.g. COLOUR is expected to be more likely to be found in a parallel structure than SIZE).

2 The data

The data for our study is a subset of the TUNA Corpus (Gatt et al., 2007), consisting of 900 references to furniture and household items, collected via a controlled experiment involving 45 participants. In addition to their TYPE, objects in the domains have COLOUR, ORIENTATION and SIZE (see Table 1). For each subset of these three attributes, there was an equal number of domains in which the minimally distinguishing description (MD) consisted of values of that subset. For example, Table 1 represents a domain in which the intended referents, $\{e_1, e_2\}$, can

```
<DESCRIPTION num='pl'>
<DESCRIPTION num='sg'>
<ATTRIBUTE name='size' value='small'>small</ATTRIBUTE>
<ATTRIBUTE name='colour' value='red'>red</ATTRIBUTE>
<ATTRIBUTE name='type' value='desk'>desk</ATTRIBUTE>
</DESCRIPTION>
and
<DESCRIPTION num='sg'>
<ATTRIBUTE name='size' value='small'>small</ATTRIBUTE>
<ATTRIBUTE name='colour' value='blue'>blue</ATTRIBUTE>
<ATTRIBUTE name='type' value='sofa'>sofa</ATTRIBUTE>
</DESCRIPTION>
</DESCRIPTION>
```

$$(\langle \textsc{size} : \mathit{small}\rangle \wedge \langle \textsc{colour} : \mathit{red}\rangle \wedge \langle \textsc{type} : \mathit{desk}\rangle) \\ \vee \\ (\langle \textsc{size} : \mathit{small}\rangle \wedge \langle \textsc{colour} : \mathit{blue}\rangle \wedge \langle \textsc{type} : \mathit{sofa}\rangle)$$

Figure 1: Corpus annotation examples

be minimally distinguished using only SIZE[2]. Thus, overspecified usage of attributes can be identified in authors' descriptions. Domain objects were randomly placed in a 3 (row) × 5 (column) grid, represented by X and Y in Table 1. These are relevant for a subset of descriptions which contain locative expressions.

Corpus descriptions are paired with an explicit XML domain representation, and annotated with semantic markup which makes clear which attributes a description contains. This markup abstracts away from differences in lexicalisation, making it an ideal resource to evaluate content determination algorithms, because it is *semantically transparent*, in the sense of this term used by van Deemter et al. (2006). This markup scheme also enables the compositional derivation of a logical form from a natural language description. For example, the XML representation of (3b) is shown in Figure 1, which also displays the LF derived from it. Each constituent NP in (3b) is annotated as a set of attributes enclosed by a DESCRIPTION tag, which is marked up as singular (sg). The two coordinates are further enclosed in a plural DESCRIPTION; correspondingly, the LF is a disjunction of (the LFs of) the two internal descriptions.

Descriptions in the corpus were elicited in 7 domains with one referent, and 13 domains with 2 referents. Plural domains represented levels of a *Value Similarity* factor. In 7 *Value-Similar* (VS) domains, referents were identifiable using identical values of the minimally distinguishing attributes. In the remaining 6 *Value-Dissimilar* (VDS) domains, the minimally distinguishing values were different. Table 1 represents a VS domain, where $\{e_1, e_2\}$ can be minimally distinguished using the same value of SIZE (*small*).

	VS		VDS	
	+Disj	−Disj	+Disj	−Disj
+aggr	20.2	15.5	2.4	3.7
−aggr	64.3	–	93.9	–
% overall	84.5	15.5	96.3	3.7

Table 2: % disjunctive and non-disjunctive plurals

In terms of our introductory discussion, referents in *Value-Similar* conditions could be minimally distinguished using a conjunction of properties, while *Value-Dissimilar* referents required a disjunction since, if two referents could be minimally distinguished by different values v and v' of an attribute A, then MD had the form $\langle \textsc{a} : v\rangle \vee \langle \textsc{a} : v'\rangle$. However, even in the VS condition, referents had different basic-level types. Thus, an author faced with a domain like Table 1 had at least the descriptive options in (3a–d). If they chose to refer to entities using basic-level values of TYPE, their description would be disjunctive (e.g. 3a). A non-disjunctive description would require the use of a superordinate value, as in (3d).

Our analysis will focus on a stratified random sample of 180 plural descriptions, referred to as $\textsc{pl}_1$, generated by taking 4 descriptions from each author (2 each from VS and VDS conditions). We also use the singular data (SG; $N = 315$). The remaining plural descriptions ($\textsc{pl}_2$; $N = 405$) are used for evaluation.

3 The logical form of plurals

Descriptions in $\textsc{pl}_1$ were first classified according to whether they were non-disjunctive (cf. 3d) or disjunctive (3a–c). The latter were further classified into aggregated (3a) and non-aggregated (3b). Table 2 displays the percentage of descriptions in each of the four categories, within each level of Value Similarity. Disjunctive descriptions were a majority in either condition, and most of these were non-aggregated. As noted in §1, these descriptions correspond to partitions of the set of referents.

Since referents in VS had identical properties except for TYPE values, the most likely reason for the majority of disjunctives in VS is that people's descriptions represented a partition of a set of referents induced by the basic-level category of the ob-

[2] TYPE was not included in the calculation of MD.

	Non-Parallel	Parallel	χ^2 ($p \leq .001$)
overspec.	24.6	75.4	92.467
underspec.	5.3	94.7	42.217
well-spec.	11	89	26

Table 3: Parallelism: % per description type

	Actual		Predicted
	$p(\text{A}, \text{SG})$	$p(\text{A}, \text{PPS})$	$p(\text{A}, \text{PPS})$
COLOUR	.680	.835	.61
SIZE	.290	.359	.28
ORIENTATION	.280	.269	.26
X-DIMENSION	.440	.517	.52
Y-DIMENSION	.630	.647	.65

Table 4: Actual and predicted usage probabilities

jects. This is strengthened by the finding that the likelihood of a description being disjunctive or non-disjunctive did not differ as a function of Value Similarity ($\chi^2 = 2.56$, $p > .1$). A χ^2 test on overall frequencies of aggregated versus non-aggregated disjunctives showed that the non-aggregated descriptions ('true' partitions) were a significant majority ($\chi^2 = 83.63$, $p < .001$). However, the greater frequency of aggregation in VS compared to VDS turned out to be significant ($\chi^2 = 15.498$, $p < .001$). Note that the predominance of non-aggregated descriptions in VS implies that properties are repeated in two disjuncts (resp. coordinate NPs), suggesting that authors are likely to redundantly propagate properties across disjuncts. This evidence goes against some recent proposals for plural reference generation which emphasise brevity (Gardent, 2002).

3.1 Conceptual gestalts and similarity

Allowing for the independent motivation for set partitioning based on TYPE values, we suggested in §1 that *parallel* descriptions such as (3b) may be more likely than *non-parallel* ones (3c), since the latter does not use the same properties to describe the two referents. Similarity, however, should also interact with attribute preferences.

For this part of the analysis, we focus exclusively on the disjunctive descriptions in PL$_1$ ($N = 150$) in both VS and VDS. The descriptions were categorised according to whether they had *parallel* or *non-parallel* semantic structure. Evidence for Similarity interacting with attribute preferences is strongest if it is found in those cases where an attribute is overspecified (i.e. used when not required for a distinguishing description). In those cases where corpus descriptions do not contain locative expressions (the X and/or Y attributes), such an overspecified usage is straightforwardly identified based on the MD of a domain. This is less straightforward in the case of locatives, since the position of objects was randomly determined in each domain. Therefore, we divided descriptions into three classes, whereby a description is considered to be:

1. *underspecified* if it does not include a locative expression and omits some MD attributes;
2. *overspecified* if either (a) it does not omit any MD attributes, but includes locatives and/or non-required visual attributes; or (b) it omits some MD attributes, but includes both a locative expression and other, non-required attributes;
3. *well-specified* otherwise.

Proportions of Parallel and Non-Parallel descriptions for each of the three classes are are shown in Table 3. In all three description types, there is an overwhelming majority of Parallel descriptions, confirmed by a χ^2 analysis. The difference in proportions of description types did not differ between VS and VDS ($\chi^2 < 1$, $p > .8$), suggesting that the tendency to redundantly repeat attributes, avoiding aggregation, is independent of whether elements of a set can be minimally distinguished using identical values.

Our second prediction was that the likelihood with which an attribute is used in a parallel structure is a function of its overall 'preference'. Thus, we expect attributes such as COLOUR to feature more than once (perhaps redundantly) in a parallel description to a greater extent than SIZE. To test this, we used the SG sample, estimating the overall probability of occurrence of a given attribute in a singular description (denoted $p(\text{A}, \text{SG})$), and using this in a non-linear regression model to predict the likelihood of usage of an attribute in a plural partitioned description with parallel semantic structure (denoted $p(\text{A}, \text{PPS})$). The data was fitted to a regression equation of the form $p(\text{A}, \text{PPS}) = k \times p(\text{A}, \text{SG})^S$. The resulting equation, shown in (4), had a near-perfect fit to the data ($R^2 = .910$)[3]. This is confirmed by comparing actual probability of occurrence in the second column of Table 4, to the predicted probabilities in the third column, which are estimated from singular probabilities using (4).

$$p(\textsc{a}, \textsc{pps}) = .713\, p(\textsc{a}, \textsc{sg})^{.912} \qquad (4)$$

Note that the probabilities in the Table confirm previous psycholinguistic findings. To the extent that probability of occurrence reflects salience and/or conceptual importance, an order over the three attributes COLOUR, SIZE and ORIENTATION can be deduced (C>>O>>S), which is compatible with the findings of Pechmann (1989), Belke and Meyer (2002) and others. The locative attributes are also ordered (Y>>X), confirming the findings of Arts (2004) that vertical location is preferred. Orderings deducible from the SG data in turn are excellent predictors of the likelihood of 'propagating' an attribute across disjuncts in a plural description, something which is likely even if an attribute is redundant, modulo the centrality or salience of the attribute in the mental gestalt corresponding to the set. Together with the earlier findings on logical form, the data evinces a dual strategy whereby (a) sets are partitioned based on basic-level conceptual category; (b) elements of the partitions are described using the same attributes if they are easily perceived and conceptualised. Thus, of the descriptions in (3) above, it is (3b) that is the norm among authors.

4 Content determination by partitioning

In this section we describe IA$_{part}$, a partitioning-based content determination algorithm. Though presented as a version of the IA, the basic strategy is generalisable beyond it. For our purposes, the assumption of a preference order will be maintained. IA$_{part}$ is distinguished from the original IA and IA$_{bool}$ (cf. §1) in two respects. First, it induces partitions opportunistically based on KB information, and this is is reflected in the way descriptions are represented. Second,, the criteria whereby a property is added to a description include a consideration of the overall salience or preference of an attribute, and its contribution to the conceptual cohesiveness of the description. Throughout the following discussion, we maintain a running example from Table 1, in which $R = \{e_1, e_2, e_5\}$.

4.1 Partitioned descriptions

IA$_{part}$ generates a *partitioned description* (D_{part}) of a set R, corresponding to a formula in Disjunctive Normal Form. D_{part} is a set of *Description Fragments* (DFs). A DF is a triple $\langle R_{\text{DF}}, T_{\text{DF}}, M_{\text{DF}}\rangle$, where $R_{\text{DF}} \subseteq R$, T_{DF} is a value of TYPE, and M_{DF} is a possibly empty set of other properties. DFs refer to disjoint subsets of R. As the representation suggests, TYPE is given a special status. IA$_{part}$ starts by selecting the basic-level values of TYPE, partitioning R and creating a DF for each element of the partition on this basis. In our example, the selection of TYPE results in two DFs, with M_{DF} initialised to empty:

(5) DF$_1$ $\langle \{e_1, e_5\}, \langle \text{TYPE} : \textit{desk}\rangle, \emptyset\rangle$
DF$_2$ $\langle \{e_2\}, \langle \text{TYPE} : \textit{sofa}\rangle, \emptyset\rangle$

Although neither DF is distinguishing, R_{DF} indicates which referents a fragment is *intended* to identify. In this way, the algorithm incorporates a 'divide-and-conquer' strategy, splitting up the referential intention into 'sub-intentions' to refer to elements of a partition. Following the initial step of selecting TYPE, the algorithm considers other properties in $\mathcal{PO}$. Suppose $\langle$COLOUR : *blue*$\rangle$ is considered first. This property is true of e_2 and e_5. Since DF$_2$ refers to e_2, the new property can be added to M_{DF_2}. Since e_5 is not the sole referent of DF$_1$, the property induces a further partitioning of this fragment, resulting in a new DF. This is identical to DF$_1$ except that it refers only to e_5 and contains $\langle$COLOUR : *blue*$\rangle$. DF$_1$ itself now refers only to e_1. Once $\langle$COLOUR : *red*$\rangle$ is considered, it is added to the latter, yielding (6).

(6) DF$_1$ $\langle \{e_1\}, \langle \text{TYPE} : \textit{desk}\rangle, \{\langle \text{COLOUR} : \textit{red}\rangle\}\rangle$
DF$_2$ $\langle \{e_2\}, \langle \text{TYPE} : \textit{sofa}\rangle, \{\langle \text{COLOUR} : \textit{blue}\rangle\}\rangle$
DF$_3$ $\langle \{e_5\}, \langle \text{TYPE} : \textit{desk}\rangle, \{\langle \text{COLOUR} : \textit{blue}\rangle\}\rangle$

The procedure $updateDescription$, which creates and updates DFs, is formalised in Algorithm 2. When some property $\langle$A : $v\rangle$ is found to be 'useful' in relation to R (in a sense to be made precise), this function is called with two arguments: $\langle$A : $v\rangle$ itself, and $R' = [\![\langle \text{A} : v\rangle]\!] \cap R$, the referents of which $\langle$A : $v\rangle$ is true. The procedure iterates through

[3] A similar analysis using linear regression gave essentially the same results.

Algorithm 2 $updateDescription(\langle \text{A} : v\rangle, R')$

```
1:  for $\langle R_{\text{DF}}, T_{\text{DF}}, M_{\text{DF}}\rangle \in D_{part}$ do
2:      if $R' = \emptyset$ then
3:          return
4:      else if $R_{\text{DF}} \subseteq R'$ then
5:          $M_{\text{DF}} \leftarrow M_{\text{DF}} \cup \{\langle \text{A} : v\rangle\}$
6:          $R' \leftarrow R' - R_{\text{DF}}$
7:      else if $R_{\text{DF}} \cap R' \neq \emptyset$ then
8:          $R_{new} \leftarrow R_{\text{DF}} \cap R'$
9:          $DF_{new} \leftarrow \langle R_{new}, T_{\text{DF}}, M_{\text{DF}} \cup \{\langle \text{A} : v\rangle\} \rangle$
10:         $D_{part} \leftarrow D_{part} \cup \{DF_{new}\}$
11:         $R_{\text{DF}} \leftarrow R_{\text{DF}} - R_{new}$
12:         $R' \leftarrow R' - R_{new}$
13:     end if
14: end for
15: if $\text{A} = \text{TYPE}$ then
16:     $D_{part} \leftarrow D_{part} \cup \{\langle R', \langle \text{A} : v\rangle, \emptyset\rangle\}$
17: else
18:     $D_{part} \leftarrow D_{part} \cup \{\langle R', \bot, \{\langle \text{A} : v\rangle\}\rangle\}$
19: end if
```

the DFs in D_{part}, adding the property to any DF such that $R_{\text{DF}} \cap R' \neq \emptyset$, until R' is empty and all referents in it have been accounted for [2.2]. As indicated in the informal discussion, there are two cases to consider for each DF:

1. $R_{\text{DF}} \subseteq R'$ [2.4]. This corresponds to our example involving $\langle$COLOUR : *blue*$\rangle$ and DF$_2$. The property is simply added to M_{DF} [2.5] and R' is updated by removing the elements thus accounted for [2.6].

2. Suppose $R_{\text{DF}} \not\subseteq R'$. If $R_{\text{DF}} \cap R'$ is empty, then $\langle \text{A} : v\rangle$ is not useful. Suppose on the other hand that $R_{\text{DF}} \cap R' \neq \emptyset$ [2.7]. This occurred with $\langle$COLOUR : *red*$\rangle$ in relation to DF$_1$. The procedure initialises R_{new}, a set holding those referents in R_{DF} which are also in R' [2.8]. A new DF (DF_{new}) is created, which is a copy of the old DF, except that (a) it contains the new property; and (b) its intended referents are R_{new} [2.9]. The new DF is included in the description [2.10], while the old DF is altered by removing R_{new} from R_{DF} [2.11]. This ensures that DFs denote disjoint subsets of R.

Two special cases arise when D_{part} is empty, or there are some elements of R' for which no DF exists. Both cases result in the construction of a new DF. An example of the former case is the initial state of the algorithm, when TYPE is added. As in example (5), the TYPE results in a new DF [2.16]. If a property is not a TYPE, the new DF has T set to null ($\bot$) and the property is included in M [2.18][4]. Note that this procedure easily generalises to the singular case, where D_{part} would only contain one DF.

4.2 Property selection criteria

IA$_{part}$'s content determination strategy maximises the similarity of a set by generating semantically parallel structures. Though contrastiveness plays a role in property selection, the 'preference' or conceptual salience of an attribute is also considered in the decision to propagate it across DFs.

Candidate properties for addition need only be true of at least one element of R. Because of the partitioning strategy, properties are not equally constrastive for all referents. For instance, in (5), e_2 needs to be distinguished from the other sofas in Table 1, while $\{e_1, e_5\}$ need to be distinguished from the desks. Therefore, distractors are held in an associative array C, such that for all $r \in R$, $C[r]$ is the set of distractors for that referent at a given stage in the procedure. Contrastiveness is defined via the following Boolean function:

$$contrastive(\langle \text{A} : v\rangle, R) \leftrightarrow \exists r \in R : \; C[r] - [\![\, \langle \text{A} : v\rangle \,]\!] \neq \emptyset \quad (7)$$

We turn next to salience and similarity. Let $A(D_{part})$ be the set of attributes included in D_{part}. A property is *salient* with respect to D_{part} if it satisfies the following:

$$salient(\langle \text{A} : v\rangle, D_{part}) \leftrightarrow \text{A} \in A(D_{part}) \wedge (.713 \, p(\text{A}, \text{SG})^{.912} > 0.5) \quad (8)$$

that is, the attribute is already included in the description, and the predicted probability of its being propagated in more than one fragment of a description is greater than chance. A potential problem arises here. Consider the description in (5) once more. At this stage, IA$_{part}$ begins to consider COLOUR. The value *red* is true of e_1, but non-contrastive (all the desks which are not in R are red). If this is the first value of COLOUR considered, (8) returns `false` because the attribute has not been used in any part of the description. On later considering $\langle$COLOUR : *blue*$\rangle$, the algorithm adds it to

[4] This only occurs if the KB is *incomplete*, that is, there some entities have no TYPE, so that R is not fully covered by the intended referents of the DFs when TYPE is initially added.

D_{part}, since it is contrastive for $\{e_2, e_5\}$, but will have failed to propagate COLOUR across fragments. As a result, IA$_{part}$ considers values of an attribute in order of *discriminatory power* (Dale, 1989), defined in the present context as follows:

$$\frac{|[\![\langle \textsc{a} : v \rangle]\!] \cap R| \; + |[\![\langle \textsc{a} : v \rangle]\!] - (U - R)|}{|[\![\langle \textsc{a} : v \rangle]\!]|} \tag{9}$$

Discriminatory power depends on the number of referents a property includes in its extension, and the number of distractors $(U - R)$ it removes. By prioritising discriminatory values, the algorithm first considers and adds $\langle$COLOUR : *blue*$\rangle$, and subsequently will include *red* because (8) returns `true`.

To continue with the example, at the stage represented by (6), only e_5 has been distinguished. ORIENTATION, the next attribute considered, is not contrastive for any referent. On considering SIZE, *small* is found to be contrastive for e_1 and e_2, and added to DF$_1$ and DF$_2$. However, SIZE is not added to DF$_3$, in spite of being present in two other fragments. This is because the probability function $p(\text{SIZE}, \text{PPS})$ returns a value below 0.5 (see Table 4, reflecting the relatively low conceptual salience of this attribute. The final description is *the blue desk, the small red desk and the small blue sofa.* This example illustrates the limits set on semantic parallelism and similarity: only attributes which are salient enough are redundantly propagated across DFs.

4.3 Complexity

An estimate of the complexity of IA$_{part}$ must account for the way properties are selected (§4.2) and the way descriptions are updated (Algorithm 2).

Property selection involves checking properties for contrastive value and salience, and updating the ordering of values of each attribute based on discriminatory power (9). Clearly, the number of times this is carried out is bounded by the number of properties in the KB, which we denote n_p. Every time a property is selected, the discriminatory power of values changes (since the number of remaining distractors changes). Now, in the worst case, all n_p properties are selected by the algorithm [5]. Each time, the algorithm must compare the remaining properties pairwise for discriminatory power, a quadratic operation with complexity $O(n_p^2)$. With respect to the procedure $updateDescription$, we need to consider the number of iterations in the for loop starting at line [2.1]. This is bounded by $n_r = |R|$ (there can be no more DFs than there are referents). Once again, if at most n_p properties are selected, then the algorithm makes at most n_r iterations n_p times, yielding complexity $O(n_p n_r)$. Overall, then, IA$_{part}$ has a worst-case runtime complexity $O(n_p^3 n_r)$.

		Mean	**Mode**	PRP
IA$_{bool}$	+ LOC	7.716	7	.7
	− LOC	8.335	7	3.5
IA$_{part}$	+ LOC	4.345	4	6.8
	− LOC	1.93	0	44.7

Table 5: Edit distance scores

5 Evaluation

IA$_{part}$ was compared to van Deemter's IA$_{bool}$ (§1) against human output in the evaluation sub-corpus PL$_2$ ($N = 405$). This was considered an adequate comparison, since IA$_{bool}$ shares with the current framework a genetic relationship with the IA. Other approaches, such as Gardent's (2002) brevity-oriented algorithm, would perform poorly on our data. As shown in §3, overspecification is extremely common in plural descriptions, suggesting that such a strategy is on the wrong track (but see §6).

IA$_{part}$ and IA$_{bool}$ were each run over the domain representation paired with each corpus description. The output logical form was compared to the LF compiled from the XML representation of an author's description (cf. Figure 1). LFs were represented as *and-or* trees, and compared using the tree edit distance algorithm of Shasha and Zhang (1990). On this measure, a value of 0 indicates identity.

Because only a subset of descriptions contain locative expressions, PL$_2$ was divided into a +LOC dataset ($N = 148$) and a −LOC dataset ($N = 257$). The preference orders for both algorithms were (C>>O>>S) for −LOC and (Y>>C>>X>>S>>O) for +LOC. These are suggested by the attribute probabilities in Table 4. Table 5 displays the mean Edit score obtained by each algorithm on the two datasets, the modal (most frequent) value, and the *perfect recall percentage* (PRP), the proportion of Edit scores of 0, indicating

[5] Only unique properties need to be considered, as each property is selected at most once, though it can be included in more than one DF.

perfect agreement with an author.

As the means and modes indicate, IA$_{part}$ outperformed IA$_{bool}$ on both datasets, with a consistently higher PRP (this coincides with the modal score in the case of $-$LOC). Pairwise $t-$tests showed that the trends were significant in both $+$LOC ($t(147) = 9.28$, $p < .001$) and $-$LOC ($t(256) = 10.039$, $p < .001$).

IA$_{bool}$ has a higher (worse) mean on $-$LOC, but a better PRP than on $+$LOC. This apparent discrepancy is partly due to variance in the edit distance scores. For instance, because the Y attribute was highest in the preference order for $+$LOC, there were occasions when both referents could be identified using the same value of Y, which was therefore included by IA$_{bool}$ at first pass, before considering disjunctions. Since Y was highly preferred by authors (see Table 4), there was higher agreement on these cases, compared to those where the values of Y were different for the two referents. In the latter case, Y was only when disjunctions were considered, if at all. The worse performance of IA$_{part}$ on $+$LOC is due to a larger choice of attributes, also resulting in greater variance, and occasionally incurring higher Edit cost when the algorithm overspecified more than a human author. This is a potential shortcoming of the partitioning strategy outlined here, when it is applied to more complex domains.

Some example outputs are given below, in a domain where COLOUR sufficed to distinguish the referents, which had different values of this attribute (i.e. an instance of the VDS condition). The formula returned by IA$_{part}$ (10a) is identical to the (LF of) the human-authored description (with Edit score of 0). The output of IA$_{bool}$ is shown in (10b).

(10) (a) $\big(\textit{fan} \wedge \textit{green}\big) \vee \big(\textit{sofa} \wedge \textit{blue}\big)$
‘the green fan and the big sofa’
(b) $\big(\textit{sofa} \vee \textit{fan}\big) \wedge \textit{small} \wedge \textit{front} \wedge \big(\textit{blue} \vee \textit{green}\big)$
‘the small, blue and green sofa and fan’

As a result of IA$_{bool}$’s requiring a property or disjunction to be true of the the entire set of referents, COLOUR is not included until disjunctions are considered, while values of SIZE and ORIENTATION are included at first pass. By contrast, IA$_{part}$ includes COLOUR before any other attribute apart from TYPE. Though overspecification is common in our data, IA$_{bool}$ overspecifies with the ‘wrong’ attributes (those which are relatively dispreferred). The rationale in IA$_{part}$ is to overspecify only if a property will enhance referent similarity, and is sufficiently salient. As for logical form, the Conjunctive Normal Form output of IA$_{bool}$ increases the Edit score, given the larger number of logical operators in (10b) compared to (10a).

6 Summary and conclusions

This paper presented a study of plural reference, showing that people (a) partition sets based on the basic level TYPE or category of their elements and (b) redundantly propagate attributes across disjuncts in a description, modulo their salience. Our algorithm partitions a set opportunistically, and incorporates a corpus-derived heuristic to estimate the salience of a property. Evaluation results showed that these principles are on the right track, with significantly better performance over a previous model (van Deemter, 2002). The partitioning strategy is related to a proposal by van Deemter and Krahmer (2007), which performs exhaustive search for a partition of a set whose elements can be described non-disjunctively. Unlike the present approach, this algorithm is non-incremental and computationally costly.

IA$_{part}$ initially performs partitioning based on the basic-level TYPE of objects, in line with the evidence. However, later partitions can be induced by other properties, possible yielding partitions even with same-TYPE referents (e.g. *the blue chair and the red chair*). Aggregation (*the blue and red chairs*) may be desirable in such cases, but limits on syntactic complexity of NPs are bound to play a role (Horacek, 2004). Another possible limitation of IA$_{part}$ is that, despite strong evidence for overspecification, complex domains could yield very lengthy outputs. Strategies to avoid them include the utilisation of other boolean operators like negation (*the desks which are not red*) (Horacek, 2004). These issues are open to future empirical research.

7 Acknowledgements

Thanks to Ehud Reiter and Ielka van der Sluis for useful comments. This work forms part of the TUNA project (`www.csd.abdn.ac.uk/research/tuna/`), supported by EPSRC grant GR/S13330/01.

References

A. Arts. 2004. *Overspecification in Instructive Texts*. Ph.D. thesis, Univiersity of Tilburg.

E. Belke and A. Meyer. 2002. Tracking the time course of multidimensional stimulus discrimination: Analysis of viewing patterns and processing times during same-different decisions. *European Journal of Cognitive Psychology*, 14(2):237–266.

R. Dale and E. Reiter. 1995. Computational interpretation of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(8):233–263.

Robert Dale. 1989. Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics, ACL-89*.

H. J. Eikmeyer and E. Ahlsèn. 1996. The cognitive process of referring to an object: A comparative study of german and swedish. In *Proceedings of the 16th Scandinavian Conference on Linguistics*.

C. Gardent, H. Manuélian, K. Striegnitz, and M. Amoia. 2004. Generating definite descriptions: Non-incrementality, inference, and data. In T. Pechman and C. Habel, editors, *Multidisciplinary Approaches to Language Production*. Mouton de Gruyter.

C. Gardent. 2002. Generating minimal definite descriptions. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL-02*.

A. Gatt, I. van der Sluis, and K. van Deemter. 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the 11th European Workshop on Natural Language Generation, ENLG-07*. To appear.

H.P. Grice. 1975. Logic and conversation. In P. Cole and J.L. Morgan, editors, *Syntax and Semantics: Speech Acts.*, volume III. Academic Press.

S. Gupta and A. J. Stent. 2005. Automatic evaluation of referring expression generation using corpora. In *Proceedings of the 1st Workshop on Using Corpora in NLG, Birmingham, UK*.

H. Horacek. 2004. On referring to sets of objects naturally. In *Proceedings of the 3rd International Conference on Natural Language Generation, INLG-04*.

P. W. Jordan and M. Walker. 2005. Learning content selection rules for generating object descriptions in dialogue. *Journal of Artificial Intelligence Research*, 24:157–194.

G. L. Murphy. 2002. *The big book of concepts*. MIT Press, Cambridge, Ma.

D. R. Olson. 1970. Language and thought: Aspects of a cognitive theory of semantics. *Psychological Review*, 77:257–273.

Thomas Pechmann. 1989. Incremental speech production and referential overspecification. *Linguistics*, 27:89–110.

I. Rock. 1983. *The Logic of Perception*. MIT Press, Cambridge, Ma.

E. Rosch, C. B. Mervis, W. Gray, D. Johnson, and P. Boyes-Braem. 1976. Basic objects in natural categories. *Cognitive Psychology*, 8:382–439.

H. Schriefers and T. Pechmann. 1988. Incremental production of referential noun phrases by human speakers. In M. Zock and G. Sabah, editors, *Advances in Natural Language Generation*, volume 1. Pinter, London.

D. Shasha and K. Zhang. 1990. Fast algorithms for unit cost editing distance between trees. *Journal of Algorithms*, 11:581–621.

K. van Deemter and E. Krahmer. 2007. Graphs and booleans: On the generation of referring expressions. In H. Bunt and R. Muskens, editors, *Computing Meaning*, volume III. Springer, Berlin.

K. van Deemter, I. van der Sluis, and A. Gatt. 2006. Building a semantically transparent corpus for the generation of referring expressions. In *Proceedings of the 4th International Conference on Natural Language Generation (Special Session on Data Sharing and Evaluation), INLG-06*.

K. van Deemter. 2002. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52.

I. van der Sluis and E. Krahmer. 2005. Towards the generation of overspecified multimodal referring expressions. In *Proceedings of the Symposium on Dialogue Modelling and Generation, 15th Annual Meeting of the Society for Text and Discourse, STD-05*.

J. Viethen and R. Dale. 2006. Algorithms for generating referring expressions: Do they do what people do? In *Proceedings of the 4th International Conference on Natural Language Generation, INLG-06*.

M. Wertheimer. 1938. Laws of organization in perceptual forms. In W. Ellis, editor, *A Source Book of Gestalt Psychology*. Routledge & Kegan Paul, London.

A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors

Joachim Wagner, Jennifer Foster, and Josef van Genabith[*]
National Centre for Language Technology
School of Computing, Dublin City University, Dublin 9, Ireland
{jwagner, jfoster, josef}@computing.dcu.ie

Abstract

This paper compares a deep and a shallow processing approach to the problem of classifying a sentence as grammatically well-formed or ill-formed. The deep processing approach uses the XLE LFG parser and English grammar: two versions are presented, one which uses the XLE directly to perform the classification, and another one which uses a decision tree trained on features consisting of the XLE's output statistics. The shallow processing approach predicts grammaticality based on n-gram frequency statistics: we present two versions, one which uses frequency thresholds and one which uses a decision tree trained on the frequencies of the rarest n-grams in the input sentence. We find that the use of a decision tree improves on the basic approach only for the deep parser-based approach. We also show that combining both the shallow and deep decision tree features is effective. Our evaluation is carried out using a large test set of grammatical and ungrammatical sentences. The ungrammatical test set is generated automatically by inserting grammatical errors into well-formed BNC sentences.

1 Introduction

This paper is concerned with the task of predicting whether a sentence contains a grammatical error. An accurate method for carrying out automatic grammaticality judgements has uses in the areas of computer-assisted language learning and grammar checking. Comparative evaluation of existing error detection approaches has been hampered by a lack of large and commonly used evaluation error corpora. We attempt to overcome this by automatically creating a large error corpus, containing four different types of frequently occurring grammatical errors. We use this corpus to evaluate the performance of two approaches to the task of automatic error detection. One approach uses low-level detection techniques based on POS n-grams. The other approach is a novel parser-based method which employs deep linguistic processing to discriminate grammatical input from ungrammatical. For both approaches, we implement a basic solution, and then attempt to improve upon this solution using a decision tree classifier. We show that combining both methods improves upon the individual methods.

N-gram-based approaches to the problem of error detection have been proposed and implemented in various forms by Atwell(1987), Bigert and Knutsson (2002), and Chodorow and Leacock (2000) amongst others. Existing approaches are hard to compare since they are evaluated on different test sets which vary in size and error density. Furthermore, most of these approaches concentrate on one type of grammatical error only, namely, context-sensitive or real-word spelling errors. We implement a vanilla n-gram-based approach which is tested on a very large test set containing four different types of error.

The idea behind the parser-based approach to error detection is to use a broad-coverage hand-crafted precision grammar to detect ungrammatical sen-

[*] Also affiliated to IBM CAS, Dublin.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 112–121, Prague, June 2007. ©2007 Association for Computational Linguistics

tences. This approach exploits the fact that a precision grammar is designed, in the traditional generative grammar sense (Chomsky, 1957), to distinguish grammatical sentences from ungrammatical sentences. This is in contrast to treebank-based grammars which tend to massively overgenerate and do not generally aim to discriminate between the two. In order for our approach to work, the coverage of the precision grammars must be broad enough to parse a large corpus of grammatical sentences, and for this reason, we choose the XLE (Maxwell and Kaplan, 1996), an efficient and robust parsing system for Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982) and the ParGram English grammar (Butt et al., 2002) for our experiments. This system employs robustness techniques, some borrowed from Optimality Theory (OT) (Prince and Smolensky, 1993), to parse extra-grammatical input (Frank et al., 1998), but crucially still distinguishes between optimal and suboptimal solutions.

The evaluation corpus is a subset of an ungrammatical version of the British National Corpus (BNC), a 100 million word balanced corpus of British English (Burnard, 2000). This corpus is obtained by automatically inserting grammatical errors into the original BNC sentences based on an analysis of a manually compiled "real" error corpus.

This paper makes the following contributions to the task of automatic error detection:

1. A novel deep processing XLE-based approach
2. An effective and novel application of decision tree machine learning to both shallow and deep approaches
3. A novel combination of deep and shallow processing
4. An evaluation of an n-gram-based approach on a wider variety of errors than has previously been carried out
5. A large evaluation error corpus

The paper is organised as follows: in Section 2, we describe previous approaches to the problem of error detection; in Section 3, a description of the error corpus used in our evaluation experiments is presented, and in Section 4, the two approaches to error detection are presented, evaluated, combined and compared. Section 5 provides a summary and suggestions for future work.

2 Background

2.1 Precision Grammars

A precision grammar is a formal grammar designed to distinguish ungrammatical from grammatical sentences. This is in contrast to large treebank-induced grammars which often accept ungrammatical input (Charniak, 1996). While high coverage is required, it is difficult to increase coverage without also increasing the amount of ungrammatical sentences that are accepted as grammatical by the grammar. Most publications in grammar-based automatic error detection focus on locating and categorising errors and giving feedback. Existing grammars are re-used (Vandeventer Faltin, 2003), or grammars of limited size are developed from scratch (Reuer, 2003).

The ParGram English LFG is a hand-crafted broad-coverage grammar developed over several years with the XLE platform (Butt et al., 2002). The XLE parser uses OT to resolve ambiguities (Prince and Smolensky, 1993). Grammar constraints resulting in rare constructions can be marked as "dispreferred" and constraints resulting in common ungrammatical constructions can be marked as "ungrammatical". The use of constraint ordering and marking increases the robustness of the grammar, while maintaining the grammatical / ungrammatical distinction (Frank et al., 1998). The English Resource Grammar (ERG) is a precision Head-Driven Phrase Structure Grammar (HPSG) of English (Copestake and Flickinger, 2000; Pollard and Sag, 1994). Its coverage is not as broad as the XLE English grammar. Baldwin et al. (2004) propose a method to identify gaps in the grammar. Blunsom and Baldwin (2006) report ongoing development.

There has been previous work using the ERG and the XLE grammars in the area of computer-assisted language learning. Bender et al. (2004) use a version of the ERG containing mal-rules to parse ill-formed sentences from the SST corpus of Japanese learner English (Emi et al., 2004). They then use the semantic representations of the ill-formed input to generate well-formed corrections. Khader et al. (2004) study whether the ParGram English LFG can be used for computer-assisted language learning by adding additional OT marks for ungrammatical constructions observed in a learner corpus. However, the evaluation is preliminary, on only 50 test items.

2.2 N-gram Methods

Most shallow approaches to grammar error detection originate from the area of real-word spelling error correction. A real-word spelling error is a spelling or typing error which results in a token which is another valid word of the language in question.

The (to our knowledge) oldest work in this area is that of Atwell (1987) who uses a POS tagger to flag POS bigrams that are unlikely according to a reference corpus. While he speculates that the bigram frequency should be compared to how often the same POS bigram is involved in errors in an error corpus, the proposed system uses the raw frequency with an empirically established threshold to decide whether a bigram indicates an error. In the same paper, a completely different approach is presented that uses the same POS tagger to consider spelling variants that have a different POS. In the example sentence *I am very **hit*** the POS of the spelling variant *hot/JJ* is added to the list NN-VB-VBD-VBN of possible POS tags of *hit*. If the POS tagger chooses *hit/JJ*, the word is flagged and the correction *hot* is proposed to the user. Unlike most n-gram-based approaches, Atwell's work aims to detect grammar errors in general and not just real-word spelling errors. However, a complete evaluation is missing.

The idea of disambiguating between the elements of confusion sets is related to word sense disambiguation. Golding (1995) builds a classifier based on a rich set of context features. Mays et al. (1991) apply the noisy channel model to the disambiguation problem. For each candidate correction S' of the input S the probability $P(S')P(S|S')$ is calculated and the most likely correction selected. This method is re-evaluated by Wilcox-O'Hearn et al. (2006) on WSJ data with artificial real-word spelling errors.

Bigert and Knutsson (2002) extend upon a basic n-gram approach by attempting to match n-grams of low frequency with similar n-grams in order to reduce overflagging. Furthermore, n-grams crossing clause boundaries are not flagged and the similarity measure is adapted in the case of phrase boundaries that usually result in low frequency n-grams.

Chodorow and Leacock (2000) use a mutual information measure in addition to raw frequency of n-grams. Apart from this, their ALEK system employs other extensions to the basic approach, for example frequency counts from both generic and word-specific corpora are used in the measures. It is not reported how much each of these contribute to the overall performance.

Rather than trying to implement all of the previous n-gram approaches, we implement the basic approach which uses rare n-grams to predict grammaticality. This property is shared by all previous shallow approaches. We also test our approach on a wider class of grammatical errors.

3 Ungrammatical Data

In this section, we discuss the notion of an artificial error corpus (Section 3.1), define the type of ungrammatical language we are dealing with (Section 3.2), and describe our procedure for creating a large artificial error corpus derived from the BNC (Section 3.3).

3.1 An Artificial Error Corpus

In order to meaningfully evaluate a shallow versus deep approach to automatic error detection, a large test set of ungrammatical sentences is needed. A corpus of ungrammatical sentences can take the form of a learner corpus (Granger, 1993; Emi et al., 2004), i. e. a corpus of sentences produced by language learners, or it can take the form of a more general error corpus comprising sentences which are not necessarily produced in a language-learning context and which contain competence and performance errors produced by native and non-native speakers of the language (Becker et al., 1999; Foster and Vogel, 2004; Foster, 2005). For both types of error corpus, it is not enough to collect a large set of sentences which are likely to contain an error - it is also necessary to examine each sentence in order to determine whether an error has actually occurred, and, if it has, to note the nature of the error. Thus, like the creation of a treebank, the creation of a corpus of ungrammatical sentences requires time and linguistic knowledge, and is by no means a trivial task.

A corpus of ungrammatical sentences which is large enough to be useful can be created automatically by inserting, deleting or replacing words in grammatical sentences. These transformations should be linguistically realistic and should, therefore, be based on an analysis of naturally produced grammatical errors. Automatically generated error corpora have been used before in natural language processing. Bigert (2004) and Wilcox-O'Hearn et al. (2006), for example, automatically introduce spelling errors into texts. Here, we generate a large error corpus by automatically inserting four different kinds of grammatical errors into BNC sentences.

3.2 Commonly Produced Grammatical Errors

Following Foster (2005), we define a sentence to be ungrammatical if all the words in the sentence are well-formed words of the language in question, but the sentence contains one or more error. This error can take the form of a performance slip which can occur due to carelessness or tiredness, or a competence error which occurs due to a lack of knowledge of a particular construction. This definition includes real-word spelling errors and excludes non-word spelling errors. It also excludes the abbreviated informal language used in electronic communication. Using the above definition as a guideline, a 20,000 word corpus of ungrammatical English sentences was collected from a variety of written texts including newspapers, academic papers, emails and website forums (Foster and Vogel, 2004; Foster, 2005). The errors in the corpus were carefully analysed and classified in terms of how they might be corrected using the three word-level correction operators: insert, delete and substitute. The following frequency ordering of the three word-level correction operators was found:
substitute (48%) > *insert* (24%) > *delete* (17%) > *combination* (11%)
Stemberger (1982) reports the same ordering of the substitution, deletion and insertion correction operators in a study of native speaker spoken language slips. Among the grammatical errors which can be corrected by substituting one word for another, the most common errors are real-word spelling errors and agreement errors. In fact, 72% of all errors fall into one of the following four classes:

1. missing word errors:
   *What **are** the subjects? > What the subjects?*
2. extra word errors:
   *Was that in the summer? > Was that in the summer **in**?*
3. real-word spelling errors:
   *She could **not** comprehend. > She could **no** comprehend.*
4. agreement errors:
   *She steered Melissa round a corner. > She steered Melissa round a **corners**.*

A similar classification was adopted by Nicholls (1999), having analysed the errors in a learner corpus. Our research is currently limited to the four error types given above, i. e. missing word errors, extra word errors, real-word spelling errors and agreements errors. However, it is possible for it to be extended to handle a wider class of errors.

3.3 Automatic Error Creation

The error creation procedure takes as input a part-of-speech-tagged corpus of sentences which are assumed to be well-formed, and outputs a corpus of ungrammatical sentences. The automatically introduced errors take the form of the four most common error types found in the manually created corpus, i. e. missing word errors, extra word errors, real-word spelling errors and agreement errors. For each sentence in the original tagged corpus, an attempt is made to automatically produce four ungrammatical sentences, one for each of the four error types. Thus, the output of the error creation procedure is, in fact, four error corpora.

3.3.1 Missing Word Errors

In the manually created error corpus of Foster (2005), missing word errors are classified based on the part-of-speech (POS) of the missing word. 98% of the missing parts-of-speech come from the following list (the frequency distribution in the error corpus is given in brackets):
det (28%) > verb (23%) > prep (21%) > pro (10%) > noun (7%) > "to" (7%) > conj (2%)
We use this information when introducing missing word errors into the BNC sentences. For each sentence, all words with the above POS tags are noted. One of these is selected and deleted. The above frequency ordering is respected so that, for example, missing determiner errors are produced more often than missing pronoun errors. No ungrammatical

sentence is produced if the original sentence contains just one word or if the sentence contains no words with parts-of-speech in the above list.

3.3.2 Extra Word Errors

We introduce extra word errors in the following three ways:

1. Random duplication of any token within a sentence: *That's the way* ***we we*** *learn here.*
2. Random duplication of any POS within a sentence: *There* ***it he*** *was.*
3. Random insertion of an arbitrary token into the sentence: *Joanna drew* ***as*** *a long breadth.*

Apart from the case of duplicate tokens, the extra words are selected from a list of tagged words compiled from a random subset of the BNC. Again, our procedure for inserting extra words is based on the analysis of extra word errors in the 20,000 word error corpus of Foster (2005).

3.3.3 Real-Word Spelling Errors

We classify an error as a real-word spelling error if it can be corrected by replacing the erroneous word with another word with a Levenshtein distance of one from the erroneous word, e.g. *the* and *they*. Based on the analysis of the manually created error corpus (Foster, 2005), we compile a list of common English real-word spelling error word pairs. For each BNC sentence, the error creation procedure records all tokens in the sentence which appear as one half of one of these word pairs. One token is selected at random and replaced by the other half of the pair. The list of common real-word spelling error pairs contains such frequently occurring words as *is* and *a*, and the procedure therefore produces an ill-formed sentence for most input sentences.

3.3.4 Agreement Errors

We introduce subject-verb and determiner-noun number agreement errors into the BNC sentences. We consider both types of agreement error equally likely and introduce the error by replacing a singular determiner, noun or verb with its plural counterpart, or vice versa. For English, subject-verb agreement errors can only be introduced for present tense verbs, and determiner-noun agreement errors can only be introduced for determiners which are marked for number, e.g. demonstratives and the indefinite article. The procedure would be more productive if applied to a morphologically richer language.

3.3.5 Covert Errors

James (1998) uses the term *covert error* to describe a genuine language error which results in a sentence which is syntactically well-formed under some interpretation different from the intended one. The prominence of covert errors in our automatically created error corpus is estimated by manually inspecting 100 sentences of each error type. The percentage of grammatical structures that are inadvertently produced for each error type and an example of each one are shown below:

- Agreement Errors, 7%
  Mary's staff ***include*** *Jones,Smith and Murphy* $>$ *Mary's staff* ***includes*** *Jones,Smith and Murphy*
- Real-Word Spelling Errors, 10%
  And ***then****?* $>$ *And* ***them****?*
- Extra Word Errors, 5%
  in defiance of the free rider prediction $>$ *in defiance of the free rider* ***near*** *prediction*
- Missing Word Errors, 13%
  She steered ***Melissa*** *round a corner* $>$ *She steered round a corner*

The occurrence of these *covert errors* can be reduced by fine-tuning the error creation procedure but they can never be completely eliminated. Indeed, they should not be eliminated from the test data, because, ideally, an optimal error detection system should be sophisticated enough to flag syntactically well-formed sentences containing covert errors as potentially ill-formed.[1]

4 Error Detection Evaluation

In this section we present the error detection evaluation experiments. The experimental setup is explained in Section 4.1, the results are presented in Section 4.2 and they are analysed in Section 4.3.

[1] An example of this is given in the XLE User Documentation (`http://www2.parc.com/isl/groups/nltt/xle/doc/`). The authors remark that an ungrammatical reading of the sentence *Lets go to the store* in which *Lets* is missing an apostrophe, is preferable to the grammatical yet implausible analysis in which *Lets* is a plural noun.

4.1 Experimental Setup

4.1.1 Test Data and Evaluation Procedure

The following steps are carried out to produce training and test data for this experiment:

1. Speech material, poems, captions and list items are removed from the BNC. 4.2 million sentences remain. The order of sentences is randomised.
2. For the purpose of cross-validation, the corpus is split into 10 parts.
3. Each part is passed to the 4 automatic error insertion modules described in Section 3.3, resulting in 40 additional sets of varying size.
4. The first 60,000 sentences of each of the 50 sets, i. e. 3 million sentences, are parsed with XLE.[2]
5. N-gram frequency information is extracted for the first 60,000 sentences of each set. An additional 20,000 is extracted as held-out data.
6. 10 sets with mixed error types are produced by joining a quarter of each respective error set.
7. For each error type (including mixed errors) and cross-validation set, the 60,000 grammatical and 60,000 ungrammatical sentences are joined.
8. Each cross-validation run uses one set out of the 10 as test data (120,000 sentences) and the remaining 9 sets for training (1,080,000 sentences).

The experiment is a standard binary classification task. The methods classify the sentences of the test sets as grammatical or ungrammatical. We use the standard measures of precision, recall, f-score and accuracy (Figure 1). True positives are understood to be ungrammatical sentences that are identified as such. The baseline precision and accuracy is 50% as half of the test data is ungrammatical. If 100% of the test data is classified as ungrammatical, recall will be 100% and f-score $2/3$. Recall shows the accuracy we would get if the grammatical half of the test data was removed. Parametrised methods are first optimised for accuracy and then the other measures are taken. Therefore, f-scores below the artificial $2/3$ baseline are meaningful.

Measure	**Formula**
precision	$tp/(tp + fp)$
recall	$tp/(tp + fn)$
f-score	$2pr * re/(pr + re)$
accuracy	$(tp + tn)/(tp + tn + fp + fn)$

Figure 1: Evaluation measures: tp = true positives, fp = false positives, tn = true negatives, fn = false negatives, pr = precision, re = recall

4.1.2 Method 1: Precision Grammar

According to the XLE documentation, a sentence is marked with a star (*) if its optimal solution uses a constraint marked as ungrammatical. We use this star feature, parser exceptions and zero number of parses to classify a sentence as ungrammatical.

4.1.3 Method 2: POS N-grams

In each cross-validation run, the full data of the remaining 9 sets of step 2 of the data generation (see Section 4.1.1) is used as a reference corpus of $0.9 \times 4,200,000 = 3,800,000$ assumedly grammatical sentences. The reference corpora and data sets are POS tagged with the IMS TreeTagger (Schmidt, 1994). Frequencies of POS n-grams ($n = 2, \ldots, 7$) are counted in the reference corpora. A test sentence is flagged as ungrammatical if it contains an n-gram below a fixed frequency threshold. Method 2 has two parameters: n and the frequency threshold.

4.1.4 Method 3: Decision Trees on XLE Output

The XLE parser outputs additional statistics for each sentence that we encode in six features:

- An integer indicating starredness (0 or 1) and various parser exceptions (-1 for time out, -2 for exceeded memory, etc.)
- The number of optimal parses[3]
- The number of unoptimal parses
- The duration of parsing
- The number of subtrees
- The number of words

[2] We use the XLE command *parse-testfile* with *parse-literally* set to 1, *max xle scratch storage* set to 1,000 MB, *timeout* to 60 seconds, and the XLE English LFG. Skimming is not switched on and fragments are.

[3] The use of preferred versus dispreferred constraints are used to distinguish optimal parses from unoptimal ones.

Training data for the decision tree learner is composed of $9 \times 60,000 = 540,000$ feature vectors from grammatical sentences and $9 \times 15,000 = 135,000$ feature vectors from ungrammatical sentences of each error type, resulting in equal amounts of grammatical and ungrammatical training data.

We choose the weka implementation of machine learning algorithms for the experiments (Witten and Frank, 2000). We use a J48 decision tree learner with the default model.

4.1.5 Method 4: Decision Trees on N-grams

Method 4 follows the setup of Method 3. However, the features are the frequencies of the rarest n-grams ($n = 2, \ldots, 7$) in the sentence. Therefore, the feature vector of one sentence contains 6 numbers.

4.1.6 Method 5: Decision Trees on Combined Feature Sets

This method combines the features of Methods 3 and 4 for training a decision tree.

4.2 Results

Table 1 shows the results for Method 1, which uses XLE starredness, parser exceptions[4] and zero parses to classify grammaticality. Table 2 shows the results for Method 2, the basic n-gram approach. Table 3 shows the results for Method 3, which classifies based on a decision tree of XLE features. The results for Method 4, the n-gram-based decision tree approach, are shown in Table 4. Finally, Table 5 shows the results for Method 5 which combines n-gram and XLE features in decision trees.

In the case of Method 2, we first have to find optimal parameters. As only very limited integer values for n and the threshold are reasonable, an exhaustive search is feasible. We considered $n = 2, \ldots, 7$ and frequency thresholds below 20,000. Separate held-out data (400,000 sentences) is used in order to avoid overfitting. Best accuracy is achieved with 5-grams and a threshold of 4. Table 2 reports results with these parameters.

Error type	**Pr.**	**Re.**	**F-Sc.**	**Acc.**
Agreement	66.2	64.6	65.4	65.8
Real-word	63.5	57.3	60.3	62.2
Extra word	64.4	59.7	62.0	63.4
Missing word	59.2	47.8	52.9	57.4
Mixed errors	63.5	57.3	60.3	62.2

Table 1: Classification results with XLE starredness, parser exceptions and zero parses (Method 1)

Error type	**Pr.**	**Re.**	**F-Sc.**	**Acc.**
Agreement	58.6	51.7	55.0	57.6
Real-word	64.0	64.9	64.5	64.2
Extra word	64.8	67.3	66.0	65.4
Missing word	57.2	48.8	52.7	56.1
Mixed errors	61.5	58.2	59.8	60.8

Table 2: Classification results with 5-gram and frequency threshold 4 (Method 2)

The standard deviation of results across cross-validation runs is below 0.006 on all measures, except for Method 4. Therefore we only report average percentages. The highest observed standard deviation is 0.0257 for recall of Method 4 on agreement errors.

For Methods 3, 4 and 5, the decision tree learner optimises accuracy and, in doing so, chooses a trade-off between precision and recall.

4.3 Analysis

Both Method 1 (Table 1) and Method 2 (Table 2) achieve above baseline accuracy for all error types. However, Method 1, which uses the XLE starred feature, parser exceptions and zero parses to determine whether or not a sentence is grammatical, slightly outperforms Method 2, which uses the fre-

Error type	**Pr.**	**Re.**	**F-Sc.**	**Acc.**
Agreement	67.0	79.3	72.6	70.1
Real-word	63.4	67.6	65.4	64.3
Extra word	63.0	66.4	64.7	63.7
Missing word	59.7	57.8	58.7	59.4
Mixed errors	63.4	67.8	65.6	64.4

Table 3: Classification results with decision tree on XLE output (Method 3)

[4] XLE parsing (see footnote 2 for configuration) runs out of time for 0.7 % and out of memory for 2.5 % of sentences, measured on training data of the first cross-validation run, i. e. 540,000 grammatical sentence and 135,000 of each error type. 14 sentences of 3 million caused the parser to terminate abnormally.

Error type	Pr.	Re.	F-Sc.	Acc.
Agreement	61.2	53.8	57.3	59.9
Real-word	65.3	64.3	64.8	65.1
Extra word	66.4	67.4	66.9	66.7
Missing word	59.1	49.2	53.7	57.5
Mixed errors	63.3	58.7	60.9	62.3

Table 4: Classification results with decision tree on vectors of frequency of rarest n-grams (Method 4)

Error type	Pr.	Re.	F-Sc.	Acc.
Agreement	67.1	75.2	70.9	69.2
Real-word	65.8	70.7	68.1	67.0
Extra word	65.9	71.2	68.5	67.2
Missing word	61.2	58.0	59.5	60.6
Mixed errors	65.2	68.8	66.9	66.0

Table 5: Classification results with decision tree on joined feature set (Method 5)

quency of POS 5-grams to detect an error. The XLE deep-processing approach is better than the n-gram-based approach for agreement errors (f-score +10.4). Examining the various types of agreement errors, we can see that this is especially the case for singular subjects followed by plural copula verbs (recall +37.7) and determiner-noun number mismatches (recall +23.6 for singular nouns and +18.0 for plural nouns), but not for plural subjects followed by singular verbs (recall -24.0). The relatively poor performance of Method 2 on agreement errors involving determiners could be due to the lack of agreement marking on the Penn Treebank determiner tag used by TreeTagger.

Method 1 is outperformed by Method 2 for real-word spelling and extra word errors (f-score -4.2, -4.0). Unsurprisingly, Method 2 has an advantage on those real-word spelling errors that change the POS (recall -8.8 for Method 1). Both methods perform poorly on missing word errors. For both methods there are only very small differences in performance between the various missing word error subtypes (identified by the POS of the deleted word).

Method 3, which uses machine learning to exploit all the information returned by the XLE parser, improves performance from Method 1, the basic XLE method, for all error types.[5] The general improvement comes from an improvement in recall, meaning that more ungrammatical sentences are actually flagged as such without compromising precision. The improvement is highest for agreement errors (f-score +7.2). Singular subject with plural copula errors (e. g. *The man are*) peak at a recall of 91.0. The Method 3 results indicate that information on the number of solutions (optimal and unoptimal), the number of subtrees, the time taken to parse the sentence and the number of words can be used to predict grammaticality. It would be interesting to investigate this approach with other parsers.

Method 4, which uses a decision tree with n-gram-based features, confirms the results of Method 2. The decision trees' root nodes are similar or even identical (depending on cross-validation run) to the decision rule of Method 2 (5-gram frequency below 4). However, the 10 decision trees have between 1,111 and 1,905 nodes and draw from all features, even bigrams and 7-grams that perform poorly on their own. The improvements are very small though and they are not significant according the criterion of non-overlapping cross-validation results. The main reason for the evaluation of Method 4 is to provide another reference point for comparison of the final method.

The overall best results are those for Method 5, the combined XLE, n-gram and machine-learning-based method, which outperforms the next best method, Method 3, on all error types apart from agreement errors (f-score -1.7, +2.7, +3.8, +0.8). For agreement errors, it seems that the relatively poor results for n-grams have a negative effect on the relatively good results for the XLE. Figure 2 shows that the performance is almost constant on ungrammatical data in the important sentence length range from 5 to 40. However, there is a negative correlation of accuracy and sentence length for grammatical sentences. Very long sentences of any kind tend to be classified as ungrammatical, except for missing word errors which remain close to the 50% baseline of coin-flipping.

For all methods, missing word errors are the worst-performing, particularly in recall (i. e. the ac-

[5] The +0.3 increase in average accuracy for extra word errors is not clearly significant as the results of cross-validation runs overlap.

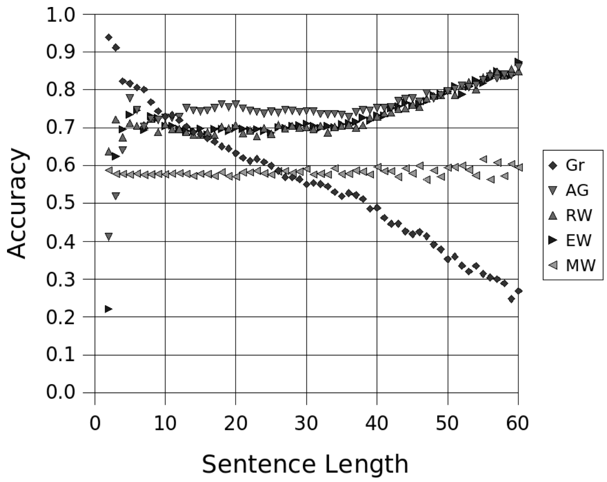


Figure 2: Accuracy by sentence length for Method 5 measured on separate grammatical and ungrammatical data: Gr = Grammatical, AG = Agreement, RW = Real-Word, EW = Extra Word, MW = Missing Word

curacy on ungrammatical data alone). This means that the omission of a word is less likely to result in the sentence being flagged as erroneous. In contrast, extra word errors perform consistently and relatively well for all methods.

5 Conclusion and Future Work

We evaluated a deep processing approach and a POS n-gram-based approach to the automatic detection of common grammatical errors in a BNC-derived artificial error corpus. The results are broken down by error type. Together with the deep approach, a decision tree machine learning algorithm can be used effectively. However, extending the shallow approach with the same learning algorithm gives only small improvements. Combining the deep and shallow approaches gives an additional improvement on all but one error type.

Our plan is to investigate why all methods perform poorly on missing word errors, to extend the error creation procedure so that it includes a wider range of errors, to try the deep approach with other parsers, to integrate additional features from state-of-the-art shallow techniques and to repeat the experiments for languages other than English.

Acknowledgements

This work is supported by the IRCSET Embark Initiative (basic research grant SC/02/298 and postdoctoral fellowship P/04/232). The training and test data used in this reseach is based on the British National Corpus (BNC), distributed by Oxford University Computing Services on behalf of the BNC Consortium. We thank Djamé Seddah for helping us to run the XLE parsing on the SFI/HEA Irish Centre for High-End Computing (ICHEC) and the authors wish to acknowledge ICHEC for the provision of computational facilities and support.

References

Eric Atwell. 1987. How to detect grammatical errors in a text without parsing it. In *Proceedings of the 3rd EACL*, pages 38–45, Morristown, NJ.

Timothy Baldwin, John Beavers, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Beauty and the beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar - and the corpus. In *Pre-Proceedings of the International Conference on Linguistic Evidence: Empirical, Theoretical and Computational Perspectives*, pages 21–26.

Markus Becker, Andrew Bredenkamp, Berthold Crysmann, and Judith Klein. 1999. Annotation of error types for German news corpus. In *Proceedings of the ATALA Workshop on Treebanks*, Paris, France.

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Timothy Baldwin. 2004. Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of the InSTIL/ICALL Symposium: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice, Italy.

Johnny Bigert and Ola Knutsson. 2002. Robust error detection: a hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proceedings RO-MAND-02*, Frascati, Italy.

Johnny Bigert. 2004. Probabilistic detection of context-sensitive spelling errors. In *Proceedings of LREC-04*, volume Five, pages 1633–1636, Lisbon, Portugal.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of EMNLP-06*, pages 164–171, Sydney.

Lou Burnard. 2000. User reference guide for the British national corpus. Technical report, Oxford University Computing Services.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7, Morristown, NJ, USA.

Eugene Charniak. 1996. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of NAACL-00*, pages 140–147, San Francisco, CA.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC-02*, Athens, Greece.

Izumi Emi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2004. The overview of the SST speech corpus of Japanese learner English and evaluation through the experiment on automatic detection of learners' errors. In *Proceedings of LREC-04*, volume Four, pages 1435–1439, Lisbon, Portugal.

Jennifer Foster and Carl Vogel. 2004. Good reasons for noting bad grammar: Constructing a corpus of ungrammatical language. In Stephan Kepser and Marga Reis, editors, *Pre-Proceedings of the International Conference on Linguistic Evidence: Empirical, Theoretical and Computational Perspectives*, pages 151–152, Tübingen, Germany.

Jennifer Foster. 2005. *Good Reasons for Noting Bad Grammar: Empirical Investigations into the Parsing of Ungrammatical Written English*. Ph.D. thesis, University of Dublin, Trinity College, Dublin, Ireland.

Anette Frank, Tracy Holloway King, Jonas Kuhn, and John Maxwell. 1998. Optimality theory style constraint ranking in large-scale LFG grammars. In *Proceedings of LFG-98*, Brisbane, Australia.

Andrew R. Golding. 1995. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 39–53, Boston, MA.

Sylviane Granger. 1993. International corpus of learner English. In J. Aarts, P. de Haan, and N.Oostdijk, editors, *English Language Corpora: Design, Analysis and Exploitation*, pages 57–71. Rodopi, Amsterdam.

Carl James. 1998. *Errors in Language Learning and Use: Exploring Error Analysis*. Addison Wesley Longman.

Ron Kaplan and Joan Bresnan. 1982. Lexical Functional Grammar: a formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press.

Rafiq Abdul Khader, Tracy Holloway King, and Miriam Butt. 2004. Deep CALL grammars: The LFG-OT experiment. http://ling.uni-konstanz.de/pages/home/butt/dgfs04call.pdf.

John Maxwell and Ron Kaplan. 1996. An Efficient Parser for LFG. In *Proceedings of LFG-96*, Grenoble.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 23(5):517–522.

D. Nicholls. 1999. The Cambridge learner corpus – error coding and analysis. In *Summer Workshop on Learner Corpora*, Tokyo, Japan.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Alan Prince and Paul Smolensky. 1993. *Optimality Theory*. MIT Press, Cambridge, Massachusetts.

Veit Reuer. 2003. *PromisD - Ein Analyseverfahren zur antizipationsfreien Erkennung und Erklärung von grammatischen Fehlern in Sprachlehrsystemen*. Ph.D. thesis, Humboldt-Universität zu Berlin, Berlin, Germany.

Helmut Schmidt. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, England.

J.P. Stemberger. 1982. Syntactic errors in speech. *Journal of Psycholinguistic Research*, 11(4):313–45.

Anne Vandeventer Faltin. 2003. *Syntactic Error Diagnosis in the context of Computer Assisted Language Learning*. Ph.D. thesis, Université de Genève.

L. Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2006. Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. http://ftp.cs.toronto.edu/pub/gh/WilcoxOHearn-etal-2006.pdf.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers.

Characterizing the Errors of Data-Driven Dependency Parsing Models

Ryan McDonald
Google Inc.
76 Ninth Avenue
New York, NY 10011
ryanmcd@google.com

Joakim Nivre
Växjö University Uppsala University
35195 Växjö 75126 Uppsala
Sweden Sweden
nivre@msi.vxu.se

Abstract

We present a comparative error analysis of the two dominant approaches in data-driven dependency parsing: global, exhaustive, graph-based models, and local, greedy, transition-based models. We show that, in spite of similar performance overall, the two models produce different types of errors, in a way that can be explained by theoretical properties of the two models. This analysis leads to new directions for parser development.

1 Introduction

Syntactic dependency representations have a long history in descriptive and theoretical linguistics and many formal models have been advanced (Hudson, 1984; Mel'čuk, 1988; Sgall et al., 1986; Maruyama, 1990). A dependency graph of a sentence represents each word and its syntactic modifiers through labeled directed arcs, as shown in Figure 1, taken from the Prague Dependency Treebank (Böhmová et al., 2003). A primary advantage of dependency representations is that they have a natural mechanism for representing discontinuous constructions, arising from long distance dependencies or free word order, through *non-projective* dependency arcs, exemplified by the arc from *jedna* to *Z* in Figure 1.

Syntactic dependency graphs have recently gained a wide interest in the computational linguistics community and have been successfully employed for many problems ranging from machine translation (Ding and Palmer, 2004) to ontology construction (Snow et al., 2004). In this work we focus on a common parsing paradigm called *data-driven dependency parsing*. Unlike grammar-based parsing, data-driven approaches learn to produce dependency graphs for sentences solely from an annotated corpus. The advantage of such models is that they are easily ported to any domain or language in which annotated resources exist.

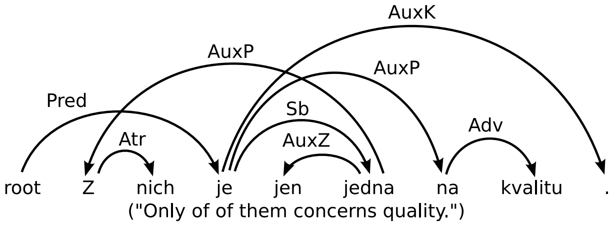


Figure 1: Example dependency graph.

As evident from the CoNLL-X shared task on dependency parsing (Buchholz and Marsi, 2006), there are currently two dominant models for data-driven dependency parsing. The first is what Buchholz and Marsi (2006) call the "all-pairs" approach, where every possible arc is considered in the construction of the optimal parse. The second is the "stepwise" approach, where the optimal parse is built stepwise and where the subset of possible arcs considered depend on previous decisions. Theoretically, these models are extremely different. The all-pairs models are globally trained, use exact (or near exact) inference algorithms, and define features over a limited history of parsing decisions. The stepwise models use local training and greedy inference algorithms, but define features over a rich history of parse decisions. However, both models obtain similar parsing accuracies

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 122–131, Prague, June 2007. ©2007 Association for Computational Linguistics

	McDonald	Nivre
Arabic	66.91	66.71
Bulgarian	87.57	87.41
Chinese	85.90	86.92
Czech	80.18	78.42
Danish	84.79	84.77
Dutch	79.19	78.59
German	87.34	85.82
Japanese	90.71	91.65
Portuguese	86.82	87.60
Slovene	73.44	70.30
Spanish	82.25	81.29
Swedish	82.55	84.58
Turkish	63.19	65.68
Overall	80.83	80.75

Table 1: Labeled parsing accuracy for top scoring systems at CoNLL-X (Buchholz and Marsi, 2006).

on a variety of languages, as seen in Table 1, which shows results for the two top performing systems in the CoNLL-X shared task, McDonald et al. (2006) ("all-pairs") and Nivre et al. (2006) ("stepwise").

Despite the similar performance in terms of overall accuracy, there are indications that the two types of models exhibit different behaviour. For example, Sagae and Lavie (2006) displayed that combining the predictions of both parsing models can lead to significantly improved accuracies. In order to pave the way for new and better methods, a much more detailed error analysis is needed to understand the strengths and weaknesses of different approaches. In this work we set out to do just that, focusing on the two top performing systems from the CoNLL-X shared task as representatives of the two dominant models in data-driven dependency parsing.

2 Two Models for Dependency Parsing

2.1 Preliminaries

Let $L = \{l_1, \ldots, l_{|L|}\}$ be a set of permissible arc labels. Let $x = w_0, w_1, \ldots, w_n$ be an input sentence where w_0=*root*. Formally, a dependency graph for an input sentence x is a labeled directed graph $G = (V, A)$ consisting of a set of nodes V and a set of labeled directed arcs $A \subseteq V \times V \times L$, i.e., if $(i, j, l) \in A$ for $i, j \in V$ and $l \in L$, then there is an arc from node i to node j with label l in the graph. A dependency graph G for sentence x must satisfy the following properties:

1. $V = \{0, 1, \ldots, n\}$
2. If $(i, j, l) \in A$, then $j \neq 0$.
3. If $(i, j, l) \in A$, then for all $i' \in V - \{i\}$ and $l' \in L$, $(i', j, l') \notin A$.
4. For all $j \in V - \{0\}$, there is a (possibly empty) sequence of nodes $i_1, \ldots, i_m \in V$ and labels $l_1, \ldots, l_m, l \in L$ such that $(0, i_1, l_1)$,(i_1, i_2, l_2), $\ldots, (i_m, j, l) \in A$.

The constraints state that the dependency graph spans the entire input (1); that the node 0 is a root (2); that each node has at most one incoming arc in the graph (3); and that the graph is connected through directed paths from the node 0 to every other node in the graph (4). A dependency graph satisfying these constraints is a directed tree originating out of the root node 0. We say that an arc (i, j, l) is *non-projective* if not all words k occurring between i and j in the linear order are dominated by i (where dominance is the transitive closure of the arc relation).

2.2 Global, Exhaustive, Graph-Based Parsing

For an input sentence, $x = w_0, w_1, \ldots, w_n$ consider the dense graph $G_x = (V_x, A_x)$ where:

1. $V_x = \{0, 1, \ldots, n\}$
2. $A_x = \{(i, j, l) \mid \forall\, i, j \in V_x \text{ and } l \in L\}$

Let $D(G_x)$ represent the subgraphs of graph G_x that are valid dependency graphs for the sentence x. Since G_x contains all possible labeled arcs, the set $D(G_x)$ must necessarily contain all valid dependency graphs for x.

Assume that there exists a dependency arc scoring function, $s : V \times V \times L \to \mathbb{R}$. Furthermore, define the score of a graph as the sum of its arc scores,

$$s(G = (V, A)) = \sum_{(i,j,l) \in A} s(i, j, l)$$

The score of a dependency arc, $s(i, j, l)$ represents the likelihood of creating a dependency from word w_i to word w_j with the label l. If the arc score function is known a priori, then the parsing problem can be stated as,

$$G = \underset{G \in D(G_x)}{\arg\max}\, s(G) = \underset{G \in D(G_x)}{\arg\max} \sum_{(i,j,l) \in A} s(i,j,l)$$

This problem is equivalent to finding the highest scoring directed spanning tree in the graph G_x originating out of the root node 0, which can be solved for both the labeled and unlabeled case in $O(n^2)$ time (McDonald et al., 2005b). In this approach, non-projective arcs are produced naturally through the inference algorithm that searches over all possible directed trees, whether projective or not.

The parsing models of McDonald work primarily in this framework. To learn arc scores, these models use large-margin structured learning algorithms (McDonald et al., 2005a), which optimize the parameters of the model to maximize the score margin between the correct dependency graph and all incorrect dependency graphs for every sentence in a training set. The learning procedure is global since model parameters are set relative to the classification of the entire dependency graph, and not just over single arc attachment decisions. The primary disadvantage of these models is that the feature representation is restricted to a limited number of graph arcs. This restriction is required so that both inference and learning are tractable.

The specific model studied in this work is that presented by McDonald et al. (2006), which factors scores over pairs of arcs (instead of just single arcs) and uses near exhaustive search for unlabeled parsing coupled with a separate classifier to label each arc. We call this system MSTParser, which is also the name of the freely available implementation.[1]

2.3 Local, Greedy, Transition-Based Parsing

A *transition system* for dependency parsing defines

1. a set C of *parser configurations*, each of which defines a (partially built) dependency graph G
2. a set T of *transitions*, each a function $t : C \rightarrow C$
3. for every sentence $x = w_0, w_1, \ldots, w_n$,
   (a) a unique *initial* configuration c_x
   (b) a set C_x of *terminal* configurations

A *transition sequence* $C_{x,m} = (c_x, c_1, \ldots, c_m)$ for a sentence x is a sequence of configurations such that $c_m \in C_x$ and, for every c_i ($c_i \neq c_x$), there is a transition $t \in T$ such that $c_i = t(c_{i-1})$. The dependency graph assigned to x by $C_{x,m}$ is the graph G_m defined by the terminal configuration c_m.

Assume that there exists a transition scoring function, $s : C \times T \rightarrow \mathbb{R}$. The score of a transition t in a configuration c, $s(c,t)$, represents the likelihood of taking transition t out of configuration c. The parsing problem consists in finding a terminal configuration $c_m \in C_x$, starting from the initial configuration c_x and taking the optimal transition $t^* = \arg\max_{t \in T} s(c,t)$ out of every configuration c. This can be seen as a greedy search for the optimal dependency graph, based on a sequence of locally optimal decisions in terms of the transition system.

Many transition systems for data-driven dependency parsing are inspired by shift-reduce parsing, where configurations contain a stack for storing partially processed nodes. Transitions in such systems add arcs to the dependency graph and/or manipulate the stack. One example is the transition system defined by Nivre (2003), which parses a sentence $x = w_0, w_1, \ldots, w_n$ in $O(n)$ time, producing a projective dependency graph satisfying conditions 1–4 in section 2.1, possibly after adding arcs $(0, i, l_r)$ for every node $i \neq 0$ that is a root in the output graph (where l_r is a special label for root modifiers). Nivre and Nilsson (2005) showed how the restriction to projective dependency graphs could be lifted by using graph transformation techniques to preprocess training data and post-process parser output, so-called *pseudo-projective parsing*.

To learn transition scores, these systems use discriminative learning methods, e.g., memory-based learning or support vector machines. The learning procedure is local since only single transitions are scored, not entire transition sequences. The primary advantage of these models is that features are not restricted to a limited number of graph arcs but can take into account the entire dependency graph built so far. The main disadvantage is that the greedy parsing strategy may lead to error propagation.

The specific model studied in this work is that presented by Nivre et al. (2006), which uses labeled pseudo-projective parsing with support vector machines. We call this system MaltParser, which is also

[1] http://mstparser.sourceforge.net

the name of the freely available implementation.[2]

2.4 Comparison

These models differ primarily with respect to three important properties.

1. **Inference:** MaltParser uses a transition-based inference algorithm that greedily chooses the best parsing decision based on a trained classifier and current parser history. MSTParser instead uses near exhaustive search over a dense graphical representation of the sentence to find the dependency graph that maximizes the score.

2. **Training:** MaltParser trains a model to make a single classification decision (choose the next transition). MSTParser trains a model to maximize the global score of correct graphs.

3. **Feature Representation:** MaltParser can introduce a rich feature history based on previous parser decisions. MSTParser is forced to restrict the score of features to a single or pair of nearby parsing decisions in order to make exhaustive inference tractable.

These differences highlight an inherent trade-off between exhaustive inference algorithms plus global learning and expressiveness of feature representations. MSTParser favors the former at the expense of the latter and MaltParser the opposite.

3 The CoNLL-X Shared Task

The CoNLL-X shared task (Buchholz and Marsi, 2006) was a large-scale evaluation of data-driven dependency parsers, with data from 13 different languages and 19 participating systems. The official evaluation metric was the *labeled attachment score* (LAS), defined as the percentage of tokens, excluding punctuation, that are assigned both the correct head and the correct dependency label.[3]

The output of all systems that participated in the shared task are available for download and constitute a rich resource for comparative error analysis. The data used in the experiments below are the outputs of MSTParser and MaltParser for all 13 languages, together with the corresponding gold standard graphs used in the evaluation. We constructed the data by simply concatenating a system's output for every language. This resulted in a single output file for each system and a corresponding single gold standard file. This method is sound because the data sets for each language contain approximately the same number of tokens – 5,000. Thus, evaluating system performance over the aggregated files can be roughly viewed as measuring system performance through an equally weighted arithmetic mean over the languages.

It could be argued that a language by language comparison would be more appropriate than comparing system performance across all languages. However, as table Table 1 shows, the difference in accuracy between the two systems is typically small for all languages, and only in a few cases is this difference significant. Furthermore, by aggregating over all languages we gain better statistical estimates of parser errors, since the data set for each individual language is very small.

4 Error Analysis

The primary purpose of this study is to characterize the errors made by standard data-driven dependency parsing models. To that end, we present a large set of experiments that relate parsing errors to a set of linguistic and structural properties of the input and predicted/gold standard dependency graphs. We argue that the results can be correlated to specific theoretical aspects of each model – in particular the trade-off highlighted in Section 2.4.

For simplicity, all experiments report labeled parsing accuracies. Identical experiments using unlabeled parsing accuracies did not reveal any additional information. Furthermore, all experiments are based on the data from all 13 languages together, as explained in section 3.

4.1 Length Factors

It is well known that parsing systems tend to have lower accuracies for longer sentences. Figure 2 shows the accuracy of both parsing models relative to sentence length (in bins of size 10: 1–10, 11–20,

[2] http://w3.msi.vxu.se/users/nivre/research/MaltParser.html

[3] In addition, results were reported for *unlabeled attachment score* (UAS) (tokens with the correct head) and *label accuracy* (LA) (tokens with the correct label).

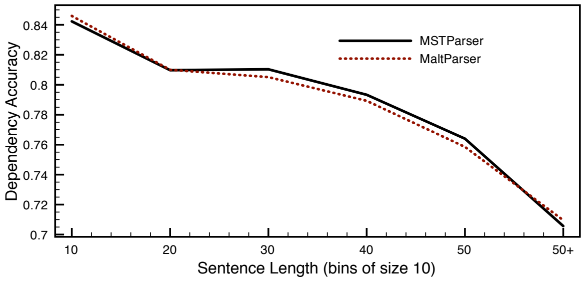


Figure 2: Accuracy relative to sentence length.

etc.). System performance is almost indistinguishable. However, MaltParser tends to perform better on shorter sentences, which require the greedy inference algorithm to make less parsing decisions. As a result, the chance of error propagation is reduced significantly when parsing these sentences. The fact that MaltParser has a higher accuracy (rather than the same accuracy) when the likelihood of error propagation is reduced comes from its richer feature representation.

Another interesting property is accuracy relative to dependency length. The length of a dependency from word w_i to word w_j is simply equal to $|i - j|$. Longer dependencies typically represent modifiers of the root or the main verb in a sentence. Shorter dependencies are often modifiers of nouns such as determiners or adjectives or pronouns modifying their direct neighbours. Figure 3 measures the precision and recall for each system relative to dependency lengths in the predicted and gold standard dependency graphs. Precision represents the percentage of predicted arcs of length d that were correct. Recall measures the percentage of gold standard arcs of length d that were correctly predicted.

Here we begin to see separation between the two systems. MSTParser is far more precise for longer dependency arcs, whereas MaltParser does better for shorter dependency arcs. This behaviour can be explained using the same reasoning as above: shorter arcs are created before longer arcs in the greedy parsing procedure of MaltParser and are less prone to error propagation. Theoretically, MSTParser should not perform better or worse for edges of any length, which appears to be the case. There is still a slight degradation, but this can be attributed to long dependencies occurring more frequently in constructions with possible ambiguity. Note that even though the area under the curve is much larger for MSTParser, the number of dependency arcs with a length greater than ten is much smaller than the number with length less than ten, which is why the overall accuracy of each system is nearly identical. For all properties considered here, bin size generally shrinks in size as the value on the x-axis increases.

4.2 Graph Factors

The structure of the predicted and gold standard dependency graphs can also provide insight into the differences between each model. For example, measuring accuracy for arcs relative to their distance to the artificial root node will detail errors at different levels of the dependency graph. For a given arc, we define this distance as the number of arcs in the reverse path from the modifier of the arc to the root. Figure 4 plots the precision and recall of each system for arcs of varying distance to the root. Precision is equal to the percentage of dependency arcs in the predicted graph that are at a distance of d and are correct. Recall is the percentage of dependency arcs in the gold standard graph that are at a distance of d and were predicted.

Figure 4 clearly shows that for arcs close to the root, MSTParser is much more precise than MaltParser, and vice-versa for arcs further away from the root. This is probably the most compelling graph given in this study since it reveals a clear distinction: MSTParser's precision degrades as the distance to the root increases whereas MaltParser's precision increases. The plots essentially run in opposite directions crossing near the middle. Dependency arcs further away from the root are usually constructed early in the parsing algorithm of MaltParser. Again a reduced likelihood of error propagation coupled with a rich feature representation benefits that parser substantially. Furthermore, MaltParser tends to overpredict root modifiers, because all words that the parser fails to attach as modifiers are automatically connected to the root, as explained in section 2.3. Hence, low precision for root modifiers (without a corresponding drop in recall) is an indication that the transition-based parser produces fragmented parses.

The behaviour of MSTParser is a little trickier to explain. One would expect that its errors should be distributed evenly over the graph. For the most part this is true, with the exception of spikes at the ends

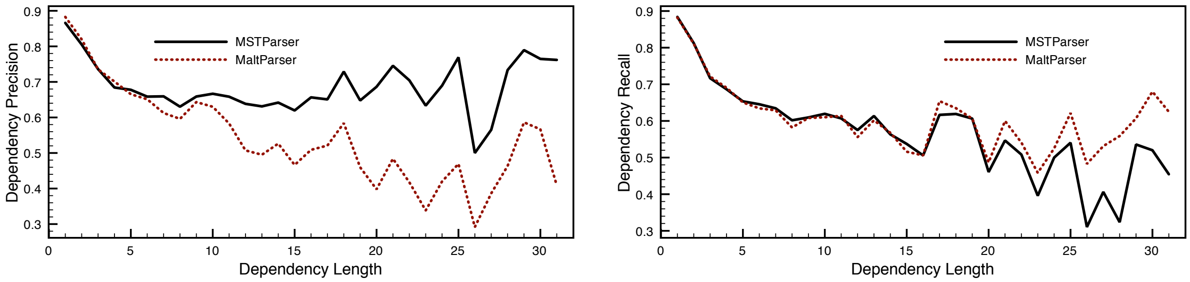


Figure 3: Dependency arc precision/recall relative to predicted/gold dependency length.

of the plot. The high performance for root modification (distance of 1) can be explained through the fact that this is typically a low entropy decision – usually the parsing algorithm has to determine the main verb from a small set of possibilities. On the other end of the plot there is a sharp downwards spike for arcs of distance greater than 10. It turns out that MSTParser over-predicts arcs near the bottom of the graph. Whereas MaltParser pushes difficult parsing decisions higher in the graph, MSTParser appears to push these decisions lower.

The next graph property we will examine aims to quantify the local neighbourhood of an arc within a dependency graph. Two dependency arcs, (i, j, l) and (i', j', l') are classified as siblings if they represent syntactic modifications of the same word, i.e., $i = i'$. Figure 5 measures the precision and recall of each system relative to the number of predicted and gold standard siblings of each arc. There is not much to distinguish between the parsers on this metric. MSTParser is slightly more precise for arcs that are predicted with more siblings, whereas MaltParser has slightly higher recall on arcs that have more siblings in the gold standard tree. Arcs closer to the root tend to have more siblings, which ties this result to the previous ones.

The final graph property we wish to look at is the degree of non-projectivity. The degree of a dependency arc from word w to word u is defined here as the number of words occurring between w and u that are not descendants of w and modify a word that does not occur between w and u (Nivre, 2006). In the example from Figure 1, the arc from *jedna* to *Z* has a degree of one, and all other arcs have a degree of zero. Figure 6 plots dependency arc precision and recall relative to arc degree in predicted and gold standard dependency graphs. MSTParser is more precise when predicting arcs with high degree and MaltParser vice-versa. Again, this can be explained by the fact that there is a tight correlation between a high degree of non-projectivity, dependency length, distance to root and number of siblings.

4.3 Linguistic Factors

It is important to relate each system's accuracy to a set of linguistic categories, such as parts of speech and dependency types. Therefore, we have made an attempt to distinguish a few broad categories that are cross-linguistically identifiable, based on the available documentation of the treebanks used in the shared task.

For parts of speech, we distinguish *verbs* (including both main verbs and auxiliaries), *nouns* (including proper names), *pronouns* (sometimes also including determiners), *adjectives*, *adverbs*, *adpositions* (prepositions, postpositions), and *conjunctions* (both coordinating and subordinating). For dependency types, we distinguish a general *root* category (for labels used on arcs from the artificial root, including either a generic label or the label assigned to predicates of main clauses, which are normally verbs), a *subject* category, an *object* category (including both direct and indirect objects), and various categories related to *coordination*.

Figure 7 shows the accuracy of the two parsers for different parts of speech. This figure measures labeled dependency accuracy relative to the part of speech of the modifier word in a dependency relation. We see that MaltParser has slightly better accuracy for nouns and pronouns, while MSTParser does better on all other categories, in particular conjunctions. This pattern is consistent with previous results insofar as verbs and conjunctions are often involved in dependencies closer to the root that span

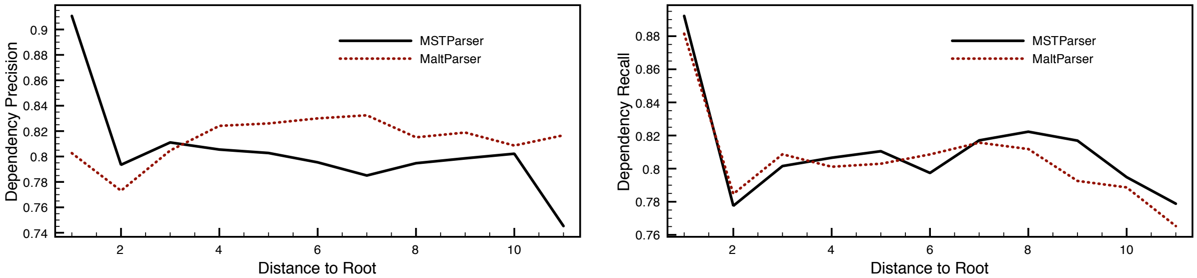


Figure 4: Dependency arc precision/recall relative to predicted/gold distance to root.

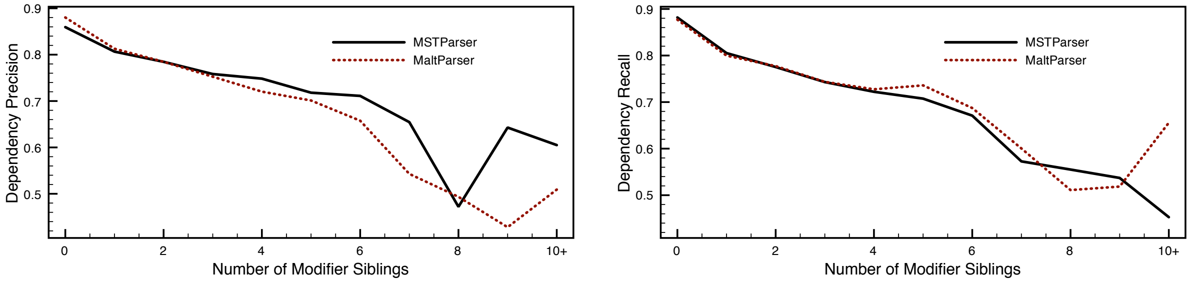


Figure 5: Dependency arc precision/recall relative to number of predicted/gold siblings.

longer distances, while nouns and pronouns are typically attached to verbs and therefore occur lower in the graph, with shorter distances. Empirically, adverbs resemble verbs and conjunctions with respect to root distance but group with nouns and pronouns for dependency length, so the former appears to be more important. In addition, both conjunctions and adverbs tend to have a high number of siblings, making the results consistent with the graph in Figure 5.

Adpositions and especially adjectives constitute a puzzle, having both high average root distance and low average dependency length. Adpositions do tend to have a high number of siblings on average, which could explain MSTParser's performance on that category. However, adjectives on average occur the furthest away from the root, have the shortest dependency length and the fewest siblings. As such, we do not have an explanation for this behaviour.

In the top half of Figure 8, we consider precision and recall for dependents of the root node (mostly verbal predicates), and for subjects and objects. As already noted, MSTParser has considerably better precision (and slightly better recall) for the root category, but MaltParser has an advantage for the nominal categories, especially subjects. A possible explanation for the latter result, in addition to the length-based and graph-based factors invoked before, is that MaltParser integrates labeling into the parsing process, so that previously assigned dependency labels can be used as features, which may be important to disambiguate subjects and objects.

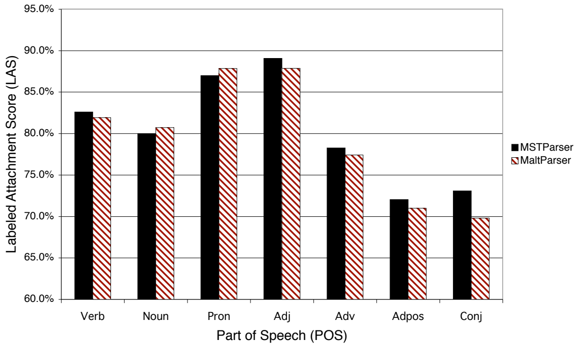


Figure 7: Accuracy for different parts of speech.

Finally, in the bottom half of Figure 8, we display precision and recall for coordinate structures, divided into different groups depending on the type of analysis adopted in a particular treebank. The category CCH (coordinating conjunction as head) contains conjunctions analyzed as heads of coordinate structures, with a special dependency label that does not describe the function of the coordinate structure in the larger syntactic structure, a type of category found in the so-called Prague style analysis of coordination and used in the data sets for Arabic, Czech,

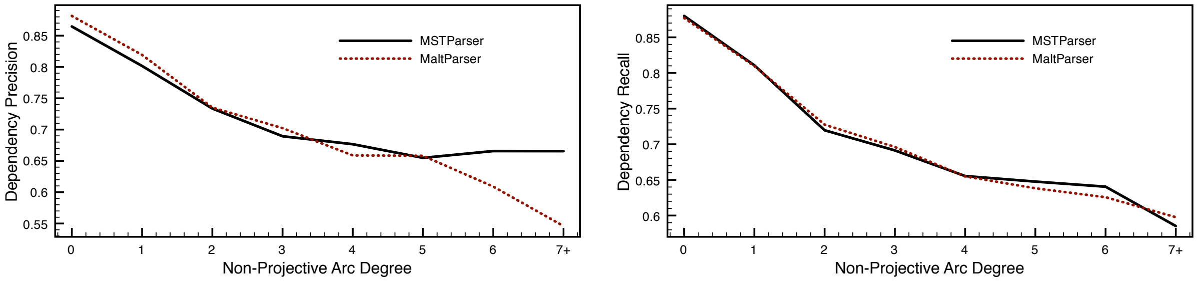


Figure 6: Dependency arc precision/recall relative to predicted/gold degree of non-projectivity.

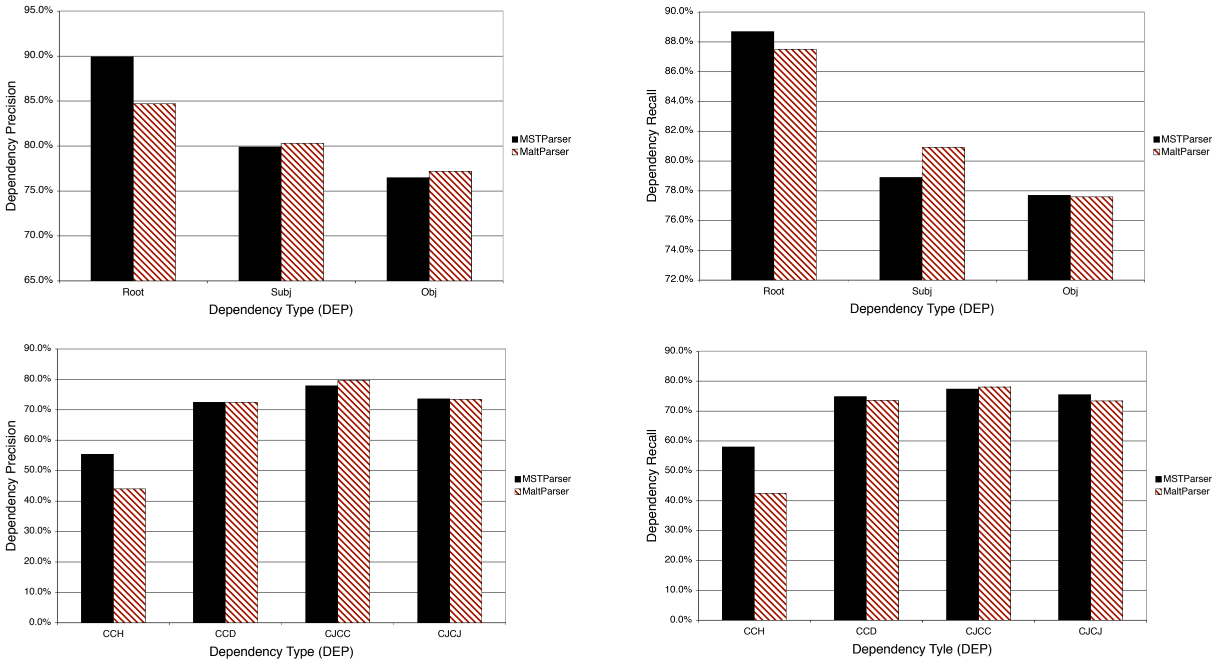


Figure 8: Precision/recall for different dependency types.

and Slovene. The category CCD (coordinating conjunction as dependent) instead denotes conjunctions that are attached as dependents of one of the conjuncts with a label that only marks them as conjunctions, a type of category found in the data sets for Bulgarian, Danish, German, Portuguese, Swedish and Turkish. The two remaining categories contain conjuncts that are assigned a dependency label that only marks them as conjuncts and that are attached either to the conjunction (CJCC) or to another conjunct (CJCJ). The former is found in Bulgarian, Danish, and German; the latter only in Portuguese and Swedish. For most of the coordination categories there is little or no difference between the two parsers, but for CCH there is a difference in both precision and recall of almost 20 percentage points to MSTParser's advantage. This can be explained by noting that, while the categories CCD, CJCC, and CJCJ denote relations that are *internal* to the coordinate structure and therefore tend to be local, the CCH relations hold between the coordinate structure and its head, which is often a relation that spans over a greater distance and is nearer the root of the dependency graph. It is likely that the difference in accuracy for this type of dependency accounts for a large part of the difference in accuracy noted earlier for conjunctions as a part of speech.

4.4 Discussion

The experiments from the previous section highlight the fundamental trade-off between global training and exhaustive inference on the one hand and expressive feature representations on the other. Error propagation is an issue for MaltParser, which typically performs worse on long sentences, long dependency arcs and arcs higher in the graphs. But this is offset by the rich feature representation available to these models that result in better decisions for frequently occurring arc types like short dependencies or subjects and objects. The errors for MSTParser are spread a little more evenly. This is expected, as the inference algorithm and feature representation should not prefer one type of arc over another.

What has been learned? It was already known that the two systems make different errors through the work of Sagae and Lavie (2006). However, in that work an arc-based voting scheme was used that took only limited account of the properties of the words connected by a dependency arc (more precisely, the overall accuracy of each parser for the part of speech of the dependent). The analysis in this work not only shows that the errors made by each system are different, but that they are different in a way that can be predicted and quantified. This is an important step in parser development.

To get some upper bounds of the improvement that can be obtained by combining the strengths of each models, we have performed two oracle experiments. Given the output of the two systems, we can envision an oracle that can optimally choose which single parse or combination of sub-parses to predict as a final parse. For the first experiment the oracle is provided with the single best parse from each system, say $G = (V, A)$ and $G' = (V', A')$. The oracle chooses a parse that has the highest number of correctly predicted labeled dependency attachments. In this situation, the oracle accuracy is 84.5%. In the second experiment the oracle chooses the tree that maximizes the number of correctly predicted dependency attachments, subject to the restriction that the tree must only contain arcs from $A \cup A'$. This can be computed by setting the weight of an arc to 1 if it is in the correct parse and in the set $A \cup A'$. All other arc weights are set to negative infinity. One can then simply find the tree that has maximal sum of arc weights using directed spanning tree algorithms. This technique is similar to the parser voting methods used by Sagae and Lavie (2006). In this situation, the oracle accuracy is 86.9%.

In both cases we see a clear increase in accuracy: 86.9% and 84.5% relative to 81% for the individual systems. This indicates that there is still potential for improvement, just by combining the two existing models. More interestingly, however, we can use the analysis to get ideas for new models. Below we sketch some possible new directions:

1. **Ensemble systems:** The error analysis presented in this paper could be used as inspiration for more refined weighting schemes for ensemble systems of the kind proposed by Sagae and Lavie (2006), making the weights depend on a range of linguistic and graph-based factors.

2. **Hybrid systems:** Rather than using an ensemble of several parsers, we may construct a single system integrating the strengths of each parser described here. This could defer to a greedy inference strategy during the early stages of the parse in order to benefit from a rich feature representation, but then default to a global exhaustive model as the likelihood for error propagation increases.

3. **Novel approaches:** The two approaches investigated are each based on a particular combination of training and inference methods. We may naturally ask what other combinations may prove fruitful. For example, what about globally trained, greedy, transition-based models? This is essentially what Daumé III et al. (2006) provide, in the form of a general search-based structured learning framework that can be directly applied to dependency parsing. The advantage of this method is that the learning can set model parameters relative to errors resulting directly from the search strategy – such as error propagation due to greedy search. When combined with MaltParser's rich feature representation, this could lead to significant improvements in performance.

5 Conclusion

We have presented a thorough study of the difference in errors made between global exhaustive graph-based parsing systems (MSTParser) and local greedy transition-based parsing systems (MaltParser). We have shown that these differences can be quantified and tied to theoretical expectations of each model, which may provide insights leading to better models in the future.

References

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: A 3-level annotation scenario. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, chapter 7. Kluwer Academic Publishers.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. CoNLL*.

Hal Daumé III, John Langford, and Daniel Marcu. 2006. Search-based structured prediction. In Submission.

Y. Ding and M. Palmer. 2004. Synchronous dependency insertion grammars: A grammar formalism for syntax based statistical MT. In *Workshop on Recent Advances in Dependency Grammars (COLING)*.

R. Hudson. 1984. *Word Grammar*. Blackwell.

H. Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proc. ACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT/EMNLP*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. CoNLL*.

I.A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. ACL*.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. CoNLL*.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. IWPT*.

J. Nivre. 2006. Constraints on non-projective dependency parsing. In *Proc. EACL*.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proc. HLT/NAACL*.

P. Sgall, E. Hajičová, and J. Panevová. 1986. *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.

R. Snow, D. Jurafsky, and A. Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Proc. NIPS*.

Probabilistic Models of Nonprojective Dependency Trees

David A. Smith
Department of Computer Science
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218 USA
dasmith@cs.jhu.edu

Noah A. Smith
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
nasmith@cs.cmu.edu

Abstract

A notable gap in research on statistical dependency parsing is a proper conditional probability distribution over nonprojective dependency trees for a given sentence. We exploit the Matrix Tree Theorem (Tutte, 1984) to derive an algorithm that efficiently sums the scores of all nonprojective trees in a sentence, permitting the definition of a conditional log-linear model over trees. While discriminative methods, such as those presented in McDonald et al. (2005b), obtain very high accuracy on standard dependency parsing tasks and can be trained and applied without marginalization, "summing trees" permits some alternative techniques of interest. Using the summing algorithm, we present competitive experimental results on four nonprojective languages, for maximum conditional likelihood estimation, minimum Bayes-risk parsing, and hidden variable training.

1 Introduction

Recently dependency parsing has received renewed interest, both in the parsing literature (Buchholz and Marsi, 2006) and in applications like translation (Quirk et al., 2005) and information extraction (Culotta and Sorensen, 2004). Dependency parsing can be used to provide a "bare bones" syntactic structure that approximates semantics, and it has the additional advantage of admitting fast parsing algorithms (Eisner, 1996; McDonald et al., 2005b) with a negligible grammar constant in many cases.

The latest state-of-the-art statistical dependency parsers are **discriminative**, meaning that they are based on classifiers trained to score trees, given a sentence, either via factored whole-structure scores (McDonald et al., 2005a) or local parsing decision scores (Hall et al., 2006). In the works cited, these scores are not intended to be interpreted as probabilistic quantities.

Here we consider weighted dependency parsing models that can be used to define well-formed conditional distributions $p(\mathbf{y} \mid \mathbf{x})$, for dependency trees $\mathbf{y}$ and a sentence $\mathbf{x}$. Conditional distributions over outputs (here, trees) given inputs (here, sentences) have certain advantages. They permit marginalization over trees to compute **posteriors** of interesting sub-events (e.g., the probability that two noun tokens bear a relation, regardless of which tree is correct). A probability model permits alternative **decoding** procedures (Goodman, 1996). Well-motivated **hidden variable** training procedures (such as EM and conditional EM) are also readily available for probabilistic models. Finally, probability models can be chained together (as in a noisy channel model), mixed, or combined in a product-of-experts.

Sequence models, context-free models, and dependency models have appeared in several guises; a cross-model comparison clarifies the contribution of this paper. First, there were generative, stochastic models like HMMs, PCFGs, and Eisner's (1996) models. Local discriminative classifiers were proposed by McCallum et al. (2000) for sequence modeling, by Ratnaparkhi et al. (1994) for constituent parsing, and by Hall et al. (2006) (among others) for

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 132–140, Prague, June 2007.

dependencies. Large-margin whole-structure models were proposed for sequence labeling by Altun et al. (2003), for constituents by Taskar et al. (2004), and for dependency trees by McDonald et al. (2005a). In this paper, we propose a model most similar to the conditional random fields—interpretable as log-linear models—of Lafferty et al. (2001), which are now widely used for sequence labeling. Log-linear models have been used in parsing by Riezler et al. (2000) (for constraint-based grammars) and Johnson (2001) and Miyao and Tsujii (2002) (for CFGs). Like McDonald et al., we use an edge-factored model that permits *nonprojective* trees; like Lafferty et al., we argue for an alternative interpretation as a **log-linear model** over structures, conditioned on the observed sentence.

In Section 2 we point out what would be required, computationally, for conditional training of nonprojective dependency models. The solution to the conditionalization problem is given in Section 3, using a widely-known but newly-applied Matrix Tree Theorem due to Tutte (1984), and experimental results are presented with a comparison to the MIRA learning algorithm used by McDonald et al. (2005a). We go on to describe and experiment with two useful applications of conditional modeling: minimum Bayes-risk decoding (Section 4) and hidden-variable training by conditional maximum likelihood estimation (Section 5). Discussion in Section 6 considers the implications of our experimental results.

Two indepedent papers, published concurrently with this one, report closely related results to ours. Koo et al. (2007) and McDonald and Satta (2007) both describe how the Matrix Tree Theorem can be applied to computing the sum of scores of edge-factored dependency trees and the edge marginals. Koo et al. compare conditional likelihood training (as here) to the averaged perceptron and a maximum margin model trained using exponentiated-gradient (Bartlett et al., 2004); the latter requires the same marginalization calculations as conditional log-linear estimation. McDonald and Satta discuss a variety of applications (including minimum Bayes-risk decoding) and give complexity results for non-edge-factored models. Interested readers are referred to those papers for further discussion.

2 Conditional Training for Nonprojective Dependency Models

Let $\mathbf{x} = \langle x_1, ..., x_n \rangle$ be a sequence of words (possibly with POS tags, lemmas, and morphological information) that are the input to a parser. $\mathbf{y}$ will refer to a directed, unlabeled dependency tree, which is a map $\mathbf{y} : \{1, ..., n\} \to \{0, ..., n\}$ from child indices to parent indices; x_0 is the invisible "wall" symbol. Let $\mathcal{Y}_\mathbf{x}$ be the set of valid dependency trees for $\mathbf{x}$. In this paper, $\mathcal{Y}_\mathbf{x}$ is equivalent to the set of all directed spanning trees over $\mathbf{x}$.[1]

A conditional model defines a family of probability distributions $p(\mathbf{y} \mid \mathbf{x})$, for all $\mathbf{x}$ and $\mathbf{y} \in \mathcal{Y}_\mathbf{x}$. We propose that this model take a log-linear form:

$$p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}) = \frac{e^{\vec{\theta} \cdot \vec{f}(\mathbf{x},\mathbf{y})}}{\sum_{\mathbf{y}' \in \mathcal{Y}_\mathbf{x}} e^{\vec{\theta} \cdot \vec{f}(\mathbf{x},\mathbf{y}')}} = \frac{e^{\vec{\theta} \cdot \vec{f}(\mathbf{x},\mathbf{y})}}{Z_{\vec{\theta}}(\mathbf{x})} \qquad (1)$$

where $\vec{f}$ is a feature vector function on parsed sentences and $\vec{\theta} \in \mathbb{R}^m$ parameterizes the model. Following McDonald et al. (2005a), we assume that the features are **edge-factored**:

$$\vec{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \vec{f}(\mathbf{x}, x_i, x_{\mathbf{y}(i)}) \qquad (2)$$

In other words, the dependencies between words in the tree are all conditionally independent of each other, given the sequence $\mathbf{x}$ and the fact that the parse is a spanning tree. Despite the constraints they impose on features, edge-factored models have the advantage of tractable $O(n^3)$ inference algorithms or, with some trickery, $O(n^2)$ maximum *a posteriori* ("best parse tree") inference algorithms in the nonprojective case. Exact nonprojective inference and estimation become intractable if we break edge factoring (McDonald and Pereira, 2006).

We wish to estimate the parameters $\vec{\theta}$ by maximizing the conditional likelihood (like a CRF) rather

[1] To be precise, every word has in-degree 1, with the sole edge pointing from the word's parent, $x_{\mathbf{y}(i)} \to x_i$. x_0 has in-degree 0. By definition, trees are acyclic. The edges need not be planar and may "cross" in the plane, since we do not have a projectivity constraint. In some formulations, exactly one node in $\mathbf{x}$ can attach to x_0; here we allow multiple nodes to attach to x_0, since this occurs with some frequency in many existing datasets. Summation over trees where x_0 has exactly one child is addressed directly by Koo et al. (2007).

than the margin (McDonald et al., 2005a). For an empirical distribution $\tilde{p}$ given by a set of training examples, this means:

$$\max_{\vec{\theta}} \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) \left(\vec{\theta}\cdot\vec{f}(\mathbf{x},\mathbf{y})\right) - \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \log Z_{\vec{\theta}}(\mathbf{x}) \tag{3}$$

This optimization problem is typically solved using a quasi-Newton numerical optimization method such as L-BFGS (Liu and Nocedal, 1989). Such a method requires the gradient of the objective function, which for θ_k is given by the following difference in expectations of the value of feature f_k:

$$\frac{\partial}{\partial\theta_k} = \mathbf{E}_{\tilde{p}(\mathbf{X},\mathbf{Y})}\left[f_k(\mathbf{X},\mathbf{Y})\right] - \mathbf{E}_{\tilde{p}(\mathbf{X})p_{\vec{\theta}}(\mathbf{Y}|\mathbf{X})}\left[f_k(\mathbf{X},\mathbf{Y})\right] \tag{4}$$

The computation of $Z_{\vec{\theta}}(\mathbf{x})$ and the sufficient statistics (second expectation in Equation 4) are typically the difficult parts. They require summing the scores of all the spanning trees for a given sentence. Note that, in large-margin training, and in standard maximum *a posteriori* decoding, only a *maximum* over spanning trees is called for—it is *conditional training* that requires $Z_{\vec{\theta}}(\mathbf{x})$. In Section 3, we will show how this can be done exactly in $O(n^3)$ time.

3 Exploiting the Matrix Tree Theorem for $Z_{\vec{\theta}}(\mathbf{x})$

We wish to apply conditional training to estimate conditional models of nonprojective trees. This requires computing $Z_{\vec{\theta}}(\mathbf{x})$ for each training example (as an inner loop to training). In this section we show how the summation can be computed and how conditional training performs.

3.1 Kirchoff Matrix

Recall that we defined the unnormalized probability (henceforth, **score**) of a dependency tree as a combination of edge-factored scores for the edges present in the tree (Eq. 2):

$$\exp \vec{\theta}\cdot\vec{f}(\mathbf{x},\mathbf{y}) = \prod_{i=1}^{n} e^{\vec{\theta}\cdot\vec{f}(\mathbf{x},x_i,x_{\mathbf{y}(i)})} = \prod_{i=1}^{n} s_{\mathbf{x},\vec{\theta}}(i,\mathbf{y}(i)) \tag{5}$$

where $\mathbf{y}(i)$ denotes the parent of x_i in $\mathbf{y}$. $s_{\mathbf{x},\vec{\theta}}(i,j)$, then, denotes the (multiplicative) contribution of the edge from child i to parent j to the total score of the tree, if the edge is present. Define the **Kirchoff matrix** $\mathbf{K}_{\mathbf{x},\vec{\theta}} \in \mathbb{R}^{n\times n}$ by

$$\left[\mathbf{K}_{\mathbf{x},\vec{\theta}}\right]_{mom,kid} = \begin{cases} -s_{\mathbf{x},\vec{\theta}}(kid,mom) & \text{if } mom \neq kid \\ \sum\limits_{j\in\{0,\ldots n\}:j\neq mom} s_{\mathbf{x},\vec{\theta}}(kid,j) & \text{if } mom = kid. \end{cases} \tag{6}$$

where mom indexes a parent node and kid a child node.

$\mathbf{K}_{\mathbf{x}\vec{\theta}}$ can be regarded as a special weighted adjacency matrix in which the ith diagonal entry is the sum of edge-scores directed into vertex i (i.e., x_i is the child)—note that the sum includes the score of attaching x_i to the wall x_0.

In our notation and in one specific form, the Matrix Tree Theorem (Tutte, 1984) states:[2]

Theorem 1 *The determinant of the Kirchoff matrix* $\mathbf{K}_{\mathbf{x},\vec{\theta}}$ *is equal to the sum of scores of all directed spanning trees in* $\mathcal{Y}_{\mathbf{x}}$ *rooted at* x_0. *Formally:*

$$\left|\mathbf{K}_{\mathbf{x},\vec{\theta}}\right| = Z_{\vec{\theta}}(\mathbf{x}).$$

A proof is omitted; see Tutte (1984).

To compute $Z_{\vec{\theta}}(\mathbf{x})$, we need only take the determinant of $\mathbf{K}_{\mathbf{x},\vec{\theta}}$, which can be done in $O(n^3)$ time using the standard LU factorization to compute the matrix inverse. Since all of the edge weights used to construct the Kirchoff matrix are positive, it is diagonally dominant and therefore non-singular (i.e., invertible).

3.2 Gradient

The gradient of $Z_{\vec{\theta}}(\mathbf{x})$ (required for numerical optimization; see Eqs. 3–4) can be efficiently computed from the same matrix inverse. While $\nabla \log Z_{\vec{\theta}}(\mathbf{x})$ equates to a vector of feature expectations (Eq. 4), we exploit instead some facts from linear algebra

[2] There are proven generalizations of this theorem (Chen, 1965; Chaiken, 1982; Minoux, 1999); we give the most specific form that applies to our case, originally proved by Tutte in 1948. Strictly speaking, our $\mathbf{K}_{\mathbf{x},\vec{\theta}}$ is not the Kirchoff matrix, but rather a *submatrix* of the Kirchoff matrix with a leftmost column of zeroes and a topmost row $[0, -s_{\mathbf{x},\vec{\theta}}(1,0), \ldots, -s_{\mathbf{x},\vec{\theta}}(n,0)]$ removed. Farther afield, Jaakkola et al. (1999) used an undirected matrix tree theorem for learning tree structures for graphical models.

$$\mathbf{K}_{\mathbf{x},\vec{\theta}} = \begin{bmatrix} \sum\limits_{j \in \{0,\ldots,n\}: j \neq 1} s_{\mathbf{x},\vec{\theta}}(1,j) & -s_{\mathbf{x},\vec{\theta}}(2,1) & \cdots & -s_{\mathbf{x},\vec{\theta}}(n,1) \\ -s_{\mathbf{x},\vec{\theta}}(1,2) & \sum\limits_{j \in \{0,\ldots,n\}: j \neq 2} s_{\mathbf{x},\vec{\theta}}(2,j) & \cdots & -s_{\mathbf{x},\vec{\theta}}(n,2) \\ \vdots & \vdots & \ddots & \vdots \\ -s_{\mathbf{x},\vec{\theta}}(1,n) & -s_{\mathbf{x},\vec{\theta}}(2,n) & \cdots & \sum\limits_{j \in \{0,\ldots,n\}: j \neq n} s_{\mathbf{x},\vec{\theta}}(n,j) \end{bmatrix}$$

and the chain rule. First, note that, for any weight θ_k,

$$\begin{aligned} & \frac{\partial \log Z_{\vec{\theta}}(\mathbf{x})}{\partial \theta_k} \\ &= \frac{\partial \log |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial \theta_k} \\ &= \frac{1}{|\mathbf{K}_{\mathbf{x},\vec{\theta}}|} \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial \theta_k} \\ &= \frac{1}{|\mathbf{K}_{\mathbf{x},\vec{\theta}}|} \sum_{i=1}^{n} \sum_{j=0}^{n} \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} \frac{\partial s_{\mathbf{x},\vec{\theta}}(i,j)}{\partial \theta_k} \\ &= \frac{1}{|\mathbf{K}_{\mathbf{x},\vec{\theta}}|} \sum_{i=1}^{n} \sum_{j=0}^{n} s_{\mathbf{x},\vec{\theta}}(i,j) f_k(\mathbf{x}, x_i, x_j) \\ & \qquad \times \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} \end{aligned} \tag{7}$$

(We assume $s_{\mathbf{x},\vec{\theta}}(i,i) = 0$, for simplicity of notation.) The last line follows from the definition of $s_{\mathbf{x},\vec{\theta}}(i,j)$ as $\exp \vec{\theta} \cdot \vec{f}(\mathbf{x}, x_i, x_j)$. Now, since $s_{\mathbf{x},\vec{\theta}}(i,j)$ affects the Kirchoff matrix in *at most* two cells—(i,i) and (j,i), the latter only when $j > 0$—we know that

$$\begin{aligned} \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} &= \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{i,i}} \frac{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{i,i}}{\partial s_{\mathbf{x},\vec{\theta}}(i,i)} \\ & \quad - \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{j,i}} \frac{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{j,i}}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} \\ &= \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{i,i}} - \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{j,i}} \end{aligned} \tag{8}$$

We have now reduced the problem of the gradient to a linear function of $\nabla |\mathbf{K}_{\mathbf{x},\vec{\theta}}|$ with respect to the cells of the matrix itself. At this point, we simplify notation and consider an arbitrary matrix $\mathbf{A}$.

The minor $m_{j,i}$ of a matrix $\mathbf{A}$ is the determinant of the submatrix obtained by striking out row j and column i of $\mathbf{A}$; the cofactor $c_{j,i}$ of $\mathbf{A}$ is then $(-1)^{i+j} m_{j,i}$. Laplace's formula defines the determinant as a linear combination of matrix cofactors of an arbitrary row j:

$$|\mathbf{A}| = \sum_{i=1}^{n} [\mathbf{A}]_{j,i} c_{j,i} \tag{9}$$

It should be clear that any $c_{j,k}$ is constant with respect to the cell $[\mathbf{A}]_{j,i}$ (since it is formed by removing row j of $\mathbf{A}$) and that other entries of $\mathbf{A}$ are constant with respect to the cell $[\mathbf{A}]_{j,i}$. Therefore:

$$\frac{\partial |\mathbf{A}|}{\partial [\mathbf{A}]_{j,i}} = c_{j,i} \tag{10}$$

The inverse matrix $\mathbf{A}^{-1}$ can also be defined in terms of cofactors:

$$[\mathbf{A}^{-1}]_{i,j} = \frac{c_{j,i}}{|\mathbf{A}|} \tag{11}$$

Combining Eqs. 10 and 11, we have:

$$\frac{\partial |\mathbf{A}|}{\partial [\mathbf{A}]_{j,i}} = |\mathbf{A}| [\mathbf{A}^{-1}]_{i,j} \tag{12}$$

Plugging back in through Eq. 8 to Eq. 7, we have:

$$\begin{aligned} \frac{\partial \log Z_{\vec{\theta}}(\mathbf{x})}{\partial \theta_k} &= \sum_{i=1}^{n} \sum_{j=0}^{n} s_{\mathbf{x},\vec{\theta}}(i,j) f_k(\mathbf{x}, x_i, x_j) \\ & \quad \times \left(\left[\mathbf{K}_{\mathbf{x},\vec{\theta}}^{-1} \right]_{i,i} - \left[\mathbf{K}_{\mathbf{x},\vec{\theta}}^{-1} \right]_{i,j} \right) \end{aligned} \tag{13}$$

where $[\mathbf{K}^{-1}]_{i,0}$ is taken to be 0. Note that the cofactors do not need to be computed directly. We proposed in Section 3.1 to get $Z_{\vec{\theta}}(\mathbf{x})$ by computing the inverse of the Kirchoff matrix (which is known to exist). Under that procedure, the marginalization is a by-product of the gradient.

decode	train	Arabic	Czech	Danish	Dutch	
m*ap*	MIRA	79.9	**81.4**	86.6	**90.0**	
	CE	80.4	80.2	**87.5**	**90.0**	(Section 3)
mBr	MIRA	79.4	80.3	85.0	87.2	(Section 4)
	CE	**80.5**	80.4	**87.5**	**90.0**	(Sections 3 & 4)

Table 1: Unlabeled dependency parsing accuracy (on test data) for two training methods (MIRA, as in McDonald et al. (2005b), and conditional estimation) and with maximum *a posteriori* (m*ap*) and minimum Bayes-risk (mBr) decoding. **Boldface** scores are best in their column on a permutation test at the .05 level.

3.3 Experiment

We compare conditional training of a nonprojective edge-factored parsing model to the online MIRA training used by McDonald et al. (2005b). Four languages with relatively common nonprojective phenomena were tested: Arabic (Hajič et al., 2004), Czech (Böhmová et al., 2003), Danish (Kromann, 2003), and Dutch (van der Beek et al., 2002). The Danish and Dutch datasets were prepared for the CoNLL 2006 shared task (Buchholz and Marsi, 2006); Arabic and Czech are from the 2007 shared task. We used the same features, extracted by McDonald's code, in both MIRA and conditional training. In this paper, we consider only *unlabeled* dependency parsing.

Our conditional training used an online gradient-based method known as stochastic gradient descent (see, e.g., Bottou, 2003). Training with MIRA and conditional estimation take about the same amount of time: approximately 50 sentences per second. Training proceeded as long as an improvement on held-out data was evident. The accuracy of the hypothesized parses for the two models, on each language, are shown in the top two rows of Tab. 1 (labeled "m*ap*" for maximum *a posteriori*, meaning that the highest-weighted tree is hypothesized).

The two methods are, not surprisingly, close in performance; conditional likelihood outperformed MIRA on Arabic and Danish, underperformed MIRA on Czech, and the two tied on Dutch. Results are significant at the .05 level on a permutation test. Conditional estimation is in practice more prone to over-fitting than maximum margin methods, though we did not see any improvement using zero-mean Gaussian priors (variance 1 or 10).

These experiments serve to validate conditional estimation as a competitive learning algorithm for parsing models, and the key contribution of the summing algorithm that permits conditional estimation.

4 Minimum Bayes-Risk Decoding

A second application of probability distributions over trees is the alternative decoding algorithm known as **minimum Bayes-risk** (mBr) decoding. The more commonly used maximum *a posteriori* decoding (also known as "Viterbi" decoding) that we applied in Section 3.3 sought to minimize the expected whole-tree loss:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}} \mathbf{E}_{p_{\vec{\theta}}(\mathbf{Y}|\mathbf{x})}\left[-\delta(\mathbf{y}, \mathbf{Y})\right] \tag{14}$$

Minimum Bayes-risk decoding generalizes this idea to an arbitrary loss function ℓ on the proposed tree:

$$\hat{\mathbf{y}} = \operatorname*{argmin}_{\mathbf{y}} \mathbf{E}_{p_{\vec{\theta}}(\mathbf{Y}|\mathbf{x})}\left[\ell(\mathbf{y}, \mathbf{Y})\right] \tag{15}$$

This technique was originally applied in speech recognition (Goel and Byrne, 2000) and translation (Kumar and Byrne, 2004); Goodman (1996) proposed a similar idea in probabilistic context-free parsing, seeking to maximize expected recall. For more applications in parsing, see Petrov and Klein (2007).

The most common loss function used to evaluate dependency parsers is the number of attachment errors, so we seek to decode using:

$$\begin{aligned} \hat{\mathbf{y}} &= \operatorname*{argmin}_{\mathbf{y}} \mathbf{E}_{p_{\vec{\theta}}(\mathbf{Y}|\mathbf{x})}\left[\sum_{i=1}^{n} -\delta(\mathbf{y}(i), \mathbf{Y}(i))\right] \\ &= \operatorname*{argmax}_{\mathbf{y}} \sum_{i=1}^{n} p_{\vec{\theta}}(\mathbf{Y}(i) = \mathbf{y}(i) \mid \mathbf{x}) \end{aligned} \tag{16}$$

To apply this decoding method, we make use of Eq. 13, which gives us the posterior probabilities of edges under the model, and the same Chiu-Liu-Edmonds maximum directed spanning tree algorithm used for maximum *a posteriori* decoding. Note that this decoding method can be applied regardless of how the model is trained. It merely requires assuming that the tree scores under the trained model (probabilistic or not) can be treated as unnormalized log-probabilities over trees given the sentence $\mathbf{x}$.

We applied minimum Bayes-risk decoding to the models trained using MIRA and using conditional estimation (see Section 3.3). Table 1 shows that, across languages, minimum Bayes-risk decoding *hurts* slightly the performance of a MIRA-trained model, but *helps* slightly or does not affect the performance of a conditionally-trained model. Since MIRA does not attempt to model the distribution over trees, this result is not surprising; interpreting weights as defining a conditional log-linear distribution is questionable under MIRA's training criterion.

One option, which we do not test here, is to use minimum Bayes-risk decoding *inside* of MIRA training, to propose a hypothesis tree (or k-best trees) at each *training* step. Doing this would more closely match the training conditions with the testing conditions; however, it is unclear whether there is a formal interpretation of such a combination, for example its relationship to McDonald et al.'s "factored MIRA."

Minimum Bayes-risk decoding, we believe, will become important in nonprojective parsing with *non-edge-factored* models. Note that minimium Bayes-risk decoding reduces *any* parsing problem to the maximum directed spanning tree problem, even if the original model is not edge-factored. All that is required are the marginals $p_{\vec{\theta}}(\mathbf{Y}(i) = \mathbf{y}(i) \mid \mathbf{x})$, which may be intractable to compute exactly, though it may be possible to develop efficient approximations.

5 Hidden Variables

A third application of probability distributions over trees is hidden-variable learning. The Expectation-Maximization (EM) algorithm (Baum and Petrie, 1966; Dempster et al., 1977; Baker, 1979), for example, is a way to maximum the likelihood of training data, marginalizing out hidden variables. This has been applied widely in unsupervised parsing (Carroll and Charniak, 1992; Klein and Manning, 2002). More recently, EM has been used to learn hidden variables in parse trees; these can be head-child annotations (Chiang and Bikel, 2002), latent head features (Matsuzaki et al., 2005; Prescher, 2005; Dreyer and Eisner, 2006), or hierarchically-split nonterminal states (Petrov et al., 2006).

To date, we know of no attempts to apply hidden variables to supervised dependency tree models. If the trees are constrained to be projective, EM is easily applied using the inside-outside variant of the parsing algorithm described by Eisner (1996) to compute the marginal probability. Moving to the nonprojective case, there are two difficulties: (a) we must marginalize over nonprojective trees and (b) we must define a generative model over $(\mathbf{X}, \mathbf{Y})$.

We have already shown in Section 3 how to solve (a); here we avoid (b) by maximizing *conditional* likelihood, marginalizing out the hidden variable, denoted $\mathbf{z}$:

$$\max_{\vec{\theta}} \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}, \mathbf{y}) \log \sum_{\mathbf{z}} p_{\vec{\theta}}(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) \qquad (17)$$

This sort of conditional training with hidden variables was carried out by Koo and Collins (2005), for example, in reranking; it is related to the information bottleneck method (Tishby et al., 1999) and contrastive estimation (Smith and Eisner, 2005).

5.1 Latent Dependency Labels

Noting that our model is edge-factored (Eq. 2), we define our hidden variables to be edge-factored as well. We can think of the hidden variables as clusters on dependency tokens, and redefine the score of an edge to be:

$$s_{\mathbf{x},\vec{\theta}}(i, j) = \sum_{z \in \mathbb{Z}} e^{\vec{\theta} \cdot \vec{f}(\mathbf{x}, x_i, x_j, z)} \qquad (18)$$

where $\mathbb{Z}$ is a set of dependency clusters.

Note that keeping the model edge-factored means that the cluster of each dependency in a tree is conditionally independent of all the others, given the words. This is computationally advantageous (we can factor out the marginalization of the hidden variable by edge), and it permits the use of any clustering method at all. For example, if an auxiliary clustering model $q(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$—perhaps one that did *not* make such independence assumptions—were used, the posterior probability $q(Z_i = z \mid \mathbf{x}, \mathbf{y})$ could be a feature in the proposed model. On the other hand, we must consider carefully the role of the dependency clusters in the model; if clusters are learned extrinsic to estimation of the parsing model, we should not expect them to be directly advantageous to parsing accuracy.

5.2 Experiments

We tried two sets of experiments with clustering. In one case, we simply augmented all of McDonald et al.'s edge features with a cluster label in hopes of improved accuracy. Models were initialized near zero, with Gaussian noise added to break symmetry among clusters.

Under these conditions, performance stayed the same or changed slightly (see Table 2); none of the improvements are significant. Note that three decoders were applied: maximum *a posteriori* (m*ap*) and minimum Bayes-risk (mBr) as described in Section 4, and "max-z," in which each possible edge was labeled and weighted only with its most likely cluster (rather than the sum over all clusters), before finding the most probable tree.[3] For each of the three languages tested, some number of clusters and some decoding method gave small improvements over the baseline.

More ambitiously, we hypothesized that many lexicalized features on edges could be "squeezed" through clusters to reduce the size of the feature set. We thus removed all word-word and lemma-lemma features and all tag fourgrams. Although this reduced our feature set by a factor of 60 or more (prior to taking a cross-product with the clusters), the damage of breaking the features was tremendous, and performance even with a thousand clusters barely broke 25% accuracy.

6 Discussion

Noting that adding latent features to nonterminals in unlexicalized context-free parsing has been very successful (Chiang and Bikel, 2002; Matsuzaki et al., 2005; Prescher, 2005; Dreyer and Eisner, 2006; Petrov et al., 2006), we were surprised not to see a more substantial performance improvement through latent features. We propose several interpretations. First, it may simply be that many more clusters may be required. Note that the label-set sizes for the labeled versions of these datasets are larger than 32 (e.g., 50 for Danish). This has the unfortunate effect of blowing up the feature space beyond the memory capacity of our machines (hence our attempts at squeezing high-dimensional features through the clusters).

# cl.	decoding	Arabic	Danish	Dutch
none	m*ap*=max-z	80.4	87.5	90.0
	mBr	80.5	87.5	90.0
2	m*ap*	80.4	87.5	89.5
	mBr	80.6	87.3	89.7
	max-z	80.4	86.3	89.4
16	m*ap*	80.4	87.6	90.1
	mBr	80.4	87.6	90.1
	max-z	80.4	87.6	90.2
32	m*ap*	80.0	87.6	–
	mBr	80.4	87.5	–
	max-z	80.0	87.5	–

Table 2: Augmenting edge features with clusters results in similar performance to conditional training with no clusters (top two lines). Scores are unlabeled dependency accuracy on test data.

Of course, improved clustering methods may also improve performance. In particular, a cluster-learning algorithm that permits clusters to split and/or merge, as in Petrov et al. (2006) or in Pereira et al. (1993), may be appropriate.

Given the relative simplicity of clustering methods for context-free parsing to date (gains were found just by using Expectation-Maximization), we believe the fundamental reason clustering was not particularly helpful here is a structural one. In context-free parsing, the latent features are (in published work to date) on nonterminal states, which are the stuctural "bridge" between context-free rules. Adding features to those states is a way of pushing information—encoded indirectly, perhaps—farther around the tree, and therefore circumventing the strict independence assumptions of probabilistic CFGs.

In an edge-factored dependency model, on the

[3] Czech experiments were not done, since the number of features (more than 14 million) was too high to multiply out by clusters.

other hand, latent features on the edges seem to have little effect. Given that they are locally "summed out" when we compute the scores of possible attachments, it should be clear that the edge clusters do not circumvent any independence assumptions. Three options appear to present themselves. First, we might attempt to learn clusters in tandem with estimating a richer, non-edge-factored model which would require approximations to $Z_{\vec{\theta}}(\mathbf{x})$, if conditional training were to be used. Note that the approximations to maximizing over spanning trees with second-order features, proposed by McDonald and Pereira (2006), do not permit estimating the clusters as part of the same process as weight estimation (at least not without modification). In the conditional estimation case, a variational approach might be appropriate. The second option is to learn clusters offline, *before* estimating the parser. (We suggested how to incorporate soft clusters into our model in Section 5.1.) This option is computationally advantageous but loses sight of the aim of learning the clusters specifically to improve parsing accuracy. Third, noting that the structural "bridge" between two coincident edges is the shared vertex (word), we might consider *word* token clustering.

We also believe this structural locality issue helps explain the modesty of the gains using minimum Bayes-risk decoding with conditional training (Section 4). In other dependency parsing scenarios, minimum Bayes-risk decoding has been found to offer significant advantages—why not here? Minimum Bayes-risk makes use of global statistical dependencies in the posterior when making local decisions. But in an edge-factored model, the edges are all conditionally independent, given that $\mathbf{y}$ is a spanning tree.

As a *post hoc* experiment, we compared purely greedy attachment (attach each word to its maximum-weighted parent, without any tree constraints). Edge scores as defined in the model were compared to minimum Bayes-risk posterior scores, and the latter were consistently better (though this always under-performed optimal spanning-tree decoding, unsurprisingly). This comparison serves only to confirm that minimum Bayes-risk decoding is a way to circumvent independence assumptions (here made by a decoder), but only when the trained model *does not* make those particular assumptions.

7 Conclusion

We have shown how to carry out exact marginalization under an edge-factored, conditional log-linear model over nonprojective dependency trees. The method has cubic runtime in the length of the sequence, but is very fast in practice. It can be used in conditional training of such a model, in minimum Bayes-risk decoding (regardless of how the model is trained), and in training with hidden variables. We demonstrated how each of these techniques gives results competitive with state-of-the-art existing dependency parsers.

Acknowledgments

The authors thank the anonymous reviewers, Jason Eisner, Keith Hall, and Sanjeev Khudanpur for helpful comments, and Michael Collins and Ryan McDonald for sharing drafts of their related, concurrent papers. This work was supported in part by NSF ITR grant IIS-0313193.

References

Y. Altun, M. Johnson, and T. Hofmann. 2003. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. of EMNLP*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Proc. of the Acoustical Society of America*, pages 547–550.

P. Bartlett, M. Collins, B. Taskar, and D. McAllester. 2004. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in NIPS 17*.

L. E. Baum and T. Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In A. Abeille, editor, *Building and Exploiting Syntactically-Annotated Corpora*. Kluwer.

L. Bottou. 2003. Stochastic learning. In *Advanced Lectures in Machine Learning*, pages 146–168. Springer.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University.

S. Chaiken. 1982. A combinatorial proof of the all minors matrix tree theorem. *SIAM Journal on Algebraic and Discrete Methods*, 3(3):319–329.

W.-K. Chen. 1965. Topological analysis for active networks. *IEEE Transactions on Circuit Theory*, 12(1):85–91.

D. Chiang and D. Bikel. 2002. Recovering latent information in treebanks. In *Proc. of COLING*.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL*.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

M. Dreyer and J. Eisner. 2006. Better informed training of latent syntactic features. In *Proc. of EMNLP*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

V. Goel and W. Byrne. 2000. Minimum Bayes risk automatic speech recognition. *Computer Speech and Language*, 14(2):115–135.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*.

J. Hajič, O. Smrž, P. Zemánek J. Šnaidauf, and E. Beška. 2004. Prague Arabic Dependency Treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*.

J. Hall, J. Nivre, and J. Nilsson. 2006. Discriminative learning for data-driven dependency parsing. In *Proc. of COLING-ACL*.

T. Jaakkola, M. Meila, and T. Jebara. 1999. Maximum entropy discrimination. In *Advances in NIPS 12*.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. of ACL*.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proc. of ACL*.

T. Koo and M. Collins. 2005. Hidden-variable models for discriminative reranking. In *Proc. of EMNLP*.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured prediction models via the Matrix-Tree Theorem. In *Proc. of EMNLP-CoNLL*.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of TLT*.

S. Kumar and W. Byrne. 2004. Minimum Bayes risk decoding for statistical machine translation. In *Proc. of HLT-NAACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

M. Minoux. 1999. A generalization of the all minors matrix tree theorem to semirings. *Discrete Mathematics*, 199:139–150.

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. of HLT*.

F. C. N. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proc. of the 31st ACL*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of HLT-NAACL*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*.

D. Prescher. 2005. Head-driven PCFGs with latent-head statistics. In *Proc. of IWPT*.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*.

A. Ratnaparkhi, S. Roukos, and R. T. Ward. 1994. A maximum entropy model for parsing. In *Proc. of ICSLP*.

S. Riezler, D. Prescher, J. Kuhn, and M. Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proc. of EMNLP*.

N. Tishby, F. C. N. Pereira, and W. Bialek. 1999. The information bottleneck method. In *Proc. of the 37th Allerton Conference on Communication, Control and Computing*, pages 368–377.

W. T. Tutte. 1984. *Graph Theory*. Addison-Wesley.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *CLIN*.

Structured Prediction Models via the Matrix-Tree Theorem

Terry Koo, Amir Globerson, Xavier Carreras and Michael Collins
MIT CSAIL, Cambridge, MA 02139, USA
{maestro,gamir,carreras,mcollins}@csail.mit.edu

Abstract

This paper provides an algorithmic framework for learning statistical models involving directed spanning trees, or equivalently non-projective dependency structures. We show how partition functions and marginals for directed spanning trees can be computed by an adaptation of Kirchhoff's Matrix-Tree Theorem. To demonstrate an application of the method, we perform experiments which use the algorithm in training both log-linear and max-margin dependency parsers. The new training methods give improvements in accuracy over perceptron-trained models.

1 Introduction

Learning with structured data typically involves searching or summing over a set with an exponential number of structured elements, for example the set of all parse trees for a given sentence. Methods for summing over such structures include the inside-outside algorithm for probabilistic context-free grammars (Baker, 1979), the forward-backward algorithm for hidden Markov models (Baum et al., 1970), and the belief-propagation algorithm for graphical models (Pearl, 1988). These algorithms compute marginal probabilities and partition functions, quantities which are central to many methods for the statistical modeling of complex structures (e.g., the EM algorithm (Baker, 1979; Baum et al., 1970), contrastive estimation (Smith and Eisner, 2005), training algorithms for CRFs (Lafferty et al., 2001), and training algorithms for max-margin models (Bartlett et al., 2004; Taskar et al., 2004a)).

This paper describes inside-outside-style algorithms for the case of directed spanning trees. These structures are equivalent to non-projective dependency parses (McDonald et al., 2005b), and more generally could be relevant to any task that involves learning a mapping from a graph to an underlying spanning tree. Unlike the case for projective dependency structures, partition functions and marginals for non-projective trees cannot be computed using dynamic-programming methods such as the inside-outside algorithm. In this paper we describe how these quantities can be computed by adapting a well-known result in graph theory: Kirchhoff's Matrix-Tree Theorem (Tutte, 1984). A naïve application of the theorem yields $O(n^4)$ and $O(n^6)$ algorithms for computation of the partition function and marginals, respectively. However, our adaptation finds the partition function and marginals in $O(n^3)$ time using simple matrix determinant and inversion operations.

We demonstrate an application of the new inference algorithm to non-projective dependency parsing. Specifically, we show how to implement two popular supervised learning approaches for this task: globally-normalized log-linear models and max-margin models. Log-linear estimation critically depends on the calculation of partition functions and marginals, which can be computed by our algorithms. For max-margin models, Bartlett et al. (2004) have provided a simple training algorithm, based on exponentiated-gradient (EG) updates, that requires computation of marginals and can thus be implemented within our framework. Both of these methods explicitly minimize the loss incurred when parsing the entire training set. This contrasts with the online learning algorithms used in previous work with spanning-tree models (McDonald et al., 2005b).

We applied the above two *marginal-based* training algorithms to six languages with varying degrees of non-projectivity, using datasets obtained from the CoNLL-X shared task (Buchholz and Marsi, 2006). Our experimental framework compared three training approaches: log-linear models, max-margin models, and the averaged perceptron. Each of these was applied to both projective and non-projective parsing. Our results demonstrate that marginal-based training yields models which out-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 141–150, Prague, June 2007. ©2007 Association for Computational Linguistics

perform those trained using the averaged perceptron.

In summary, the contributions of this paper are:

1. We introduce algorithms for *inside-outside-style* calculations for directed spanning trees, or equivalently non-projective dependency structures. These algorithms should have wide applicability in learning problems involving spanning-tree structures.

2. We illustrate the utility of these algorithms in log-linear training of dependency parsing models, and show improvements in accuracy when compared to averaged-perceptron training.

3. We also train max-margin models for dependency parsing via an EG algorithm (Bartlett et al., 2004). The experiments presented here constitute the first application of this algorithm to a large-scale problem. We again show improved performance over the perceptron.

The goal of our experiments is to give a rigorous comparative study of the marginal-based training algorithms and a highly-competitive baseline, the averaged perceptron, using the same feature sets for all approaches. We stress, however, that the purpose of this work is not to give competitive performance on the CoNLL data sets; this would require further engineering of the approach.

Similar adaptations of the Matrix-Tree Theorem have been developed independently and simultaneously by Smith and Smith (2007) and McDonald and Satta (2007); see Section 5 for more discussion.

2 Background

2.1 Discriminative Dependency Parsing

Dependency parsing is the task of mapping a sentence $\mathbf{x}$ to a dependency structure y. Given a sentence $\mathbf{x}$ with n words, a dependency for that sentence is a tuple (h, m) where $h \in [0 \ldots n]$ is the index of the head word in the sentence, and $m \in [1 \ldots n]$ is the index of a modifier word. The value $h = 0$ is a special *root-symbol* that may only appear as the head of a dependency. We use $\mathcal{D}(\mathbf{x})$ to refer to all possible dependencies for a sentence $\mathbf{x}$: $\mathcal{D}(\mathbf{x}) = \{(h, m) : h \in [0 \ldots n], m \in [1 \ldots n]\}$.

A dependency parse is a set of dependencies that forms a directed tree, with the sentence's root-symbol as its root. We will consider both *projective* trees, where dependencies are not allowed to cross, and *non-projective* trees, where crossing dependencies are allowed. Dependency annotations for some languages, for example Czech, can exhibit a significant number of crossing dependencies. In addition, we consider both *single-root* and *multi-root* trees. In a single-root tree y, the root-symbol has exactly one child, while in a multi-root tree, the root-symbol has one or more children. This distinction is relevant as our training sets include both single-root corpora (in which all trees are single-root structures) and multi-root corpora (in which some trees are multi-root structures).

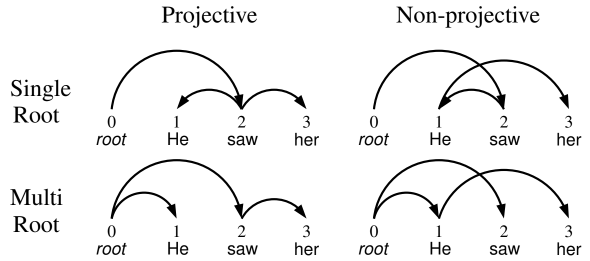


Figure 1: Examples of the four types of dependency structures. We draw dependency arcs from head to modifier.

The two distinctions described above are orthogonal, yielding four classes of dependency structures; see Figure 1 for examples of each kind of structure. We use $\mathcal{T}_p^s(\mathbf{x})$ to denote the set of all possible projective single-root dependency structures for a sentence $\mathbf{x}$, and $\mathcal{T}_{np}^s(\mathbf{x})$ to denote the set of single-root non-projective structures for $\mathbf{x}$. The sets $\mathcal{T}_p^m(\mathbf{x})$ and $\mathcal{T}_{np}^m(\mathbf{x})$ are defined analogously for multi-root structures. In contexts where any class of dependency structures may be used, we use the notation $\mathcal{T}(\mathbf{x})$ as a placeholder that may be defined as $\mathcal{T}_p^s(\mathbf{x})$, $\mathcal{T}_{np}^s(\mathbf{x})$, $\mathcal{T}_p^m(\mathbf{x})$ or $\mathcal{T}_{np}^m(\mathbf{x})$.

Following McDonald et al. (2005a), we use a discriminative model for dependency parsing. Features in the model are defined through a function $\mathbf{f}(\mathbf{x}, h, m)$ which maps a sentence $\mathbf{x}$ together with a dependency (h, m) to a feature vector in $\mathbb{R}^d$. A feature vector can be sensitive to any properties of the triple $(\mathbf{x}, h, m)$. Given a parameter vector $\mathbf{w}$, the optimal dependency structure for a sentence $\mathbf{x}$ is

$$y^*(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{T}(\mathbf{x})} \sum_{(h,m) \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m) \quad (1)$$

where the set $\mathcal{T}(\mathbf{x})$ can be defined as $\mathcal{T}_p^s(\mathbf{x})$, $\mathcal{T}_{np}^s(\mathbf{x})$, $\mathcal{T}_p^m(\mathbf{x})$ or $\mathcal{T}_{np}^m(\mathbf{x})$, depending on the type of parsing.

The parameters $\mathbf{w}$ will be learned from a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where each $\mathbf{x}_i$ is a sentence and each y_i is a dependency structure. Much of the previous work on learning $\mathbf{w}$ has focused on training local models (see Section 5). McDonald et al. (2005a; 2005b) trained global models using online algorithms such as the perceptron algorithm or MIRA. In this paper we consider training algorithms based on work in conditional random fields (CRFs) (Lafferty et al., 2001) and max-margin methods (Taskar et al., 2004a).

2.2 Three Inference Problems

This section highlights three inference problems which arise in training and decoding discriminative dependency parsers, and which are central to the approaches described in this paper.

Assume that we have a vector $\boldsymbol{\theta}$ with values $\theta_{h,m} \in \mathbb{R}$ for all $(h, m) \in \mathcal{D}(\mathbf{x})$; these values correspond to weights on the different dependencies in $\mathcal{D}(\mathbf{x})$. Define a conditional distribution over all dependency structures $y \in \mathcal{T}(\mathbf{x})$ as follows:

$$P(y \,|\, \mathbf{x}; \boldsymbol{\theta}) = \frac{\exp\left\{\sum_{(h,m)\in y} \theta_{h,m}\right\}}{Z(\mathbf{x}; \boldsymbol{\theta})} \quad (2)$$

$$Z(\mathbf{x}; \boldsymbol{\theta}) = \sum_{y\in\mathcal{T}(\mathbf{x})} \exp\left\{\sum_{(h,m)\in y} \theta_{h,m}\right\} \quad (3)$$

The function $Z(\mathbf{x}; \boldsymbol{\theta})$ is commonly referred to as the *partition function*.

Given the distribution $P(y \,|\, \mathbf{x}; \boldsymbol{\theta})$, we can define the *marginal probability* of a dependency (h, m) as

$$\mu_{h,m}(\mathbf{x}; \boldsymbol{\theta}) = \sum_{y\in\mathcal{T}(\mathbf{x})\,:\,(h,m)\in y} P(y \,|\, \mathbf{x}; \boldsymbol{\theta})$$

The inference problems are then as follows:

Problem 1: **Decoding:**
Find $\mathrm{argmax}_{y\in\mathcal{T}(\mathbf{x})} \sum_{(h,m)\in y} \theta_{h,m}$

Problem 2: **Computation of the Partition Function:** Calculate $Z(\mathbf{x}; \boldsymbol{\theta})$.

Problem 3: **Computation of the Marginals:**
For all $(h, m) \in \mathcal{D}(\mathbf{x})$, calculate $\mu_{h,m}(\mathbf{x}; \boldsymbol{\theta})$.

Note that all three problems require a maximization or summation over the set $\mathcal{T}(\mathbf{x})$, which is exponential in size. There is a clear motivation for being able to solve Problem 1: by setting $\theta_{h,m} = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$, the optimal dependency structure $y^*(\mathbf{x}; \mathbf{w})$ (see Eq. 1) can be computed. In this paper the motivation for solving Problems 2 and 3 arises from training algorithms for discriminative models. As we will describe in Section 4, both log-linear and max-margin models can be trained via methods that make direct use of algorithms for Problems 2 and 3.

In the case of projective dependency structures (i.e., $\mathcal{T}(\mathbf{x})$ defined as $\mathcal{T}_p^s(\mathbf{x})$ or $\mathcal{T}_p^m(\mathbf{x})$), there are well-known algorithms for all three inference problems. Decoding can be carried out using Viterbi-style dynamic-programming algorithms, for example the $O(n^3)$ algorithm of Eisner (1996). Computation of the marginals and partition function can also be achieved in $O(n^3)$ time, using a variant of the inside-outside algorithm (Baker, 1979) applied to the Eisner (1996) data structures (Paskin, 2001).

In the non-projective case (i.e., $\mathcal{T}(\mathbf{x})$ defined as $\mathcal{T}_{np}^s(\mathbf{x})$ or $\mathcal{T}_{np}^m(\mathbf{x})$), McDonald et al. (2005b) describe how the CLE algorithm (Chu and Liu, 1965; Edmonds, 1967) can be used for decoding. However, it is not possible to compute the marginals and partition function using the inside-outside algorithm. We next describe a method for computing these quantities in $O(n^3)$ time using matrix inverse and determinant operations.

3 Spanning-tree inference using the Matrix-Tree Theorem

In this section we present algorithms for computing the partition function and marginals, as defined in Section 2.2, for non-projective parsing. We first reiterate the observation of McDonald et al. (2005a) that non-projective parses correspond to directed spanning trees on a complete directed graph of n nodes, where n is the length of the sentence. The above inference problems thus involve summation over the set of all directed spanning trees. Note that this set is exponentially large, and there is no obvious method for decomposing the sum into dynamic-programming-like subproblems. This section describes how a variant of Kirchhoff's Matrix-Tree Theorem (Tutte, 1984) can be used to evaluate the partition function and marginals efficiently.

In what follows, we consider the single-root setting (i.e., $\mathcal{T}(\mathbf{x}) = \mathcal{T}_{np}^s(\mathbf{x})$), leaving the multi-root case (i.e., $\mathcal{T}(\mathbf{x}) = \mathcal{T}^m_{np}(\mathbf{x})$) to Section 3.3. For a sentence $\mathbf{x}$ with n words, define a complete directed graph G on n nodes, where each node corresponds to a word in $\mathbf{x}$, and each edge corresponds to a dependency between two words in $\mathbf{x}$. Note that G does *not* include the root-symbol $h = 0$, nor does it account for any dependencies $(0, m)$ headed by the root-symbol. We assign non-negative weights to the edges of this graph, yielding the following weighted adjacency matrix $A(\boldsymbol{\theta}) \in \mathbb{R}^{n \times n}$, for $h, m = 1 \ldots n$:

$$A_{h,m}(\boldsymbol{\theta}) \;=\; \begin{cases} 0, & \text{if } h = m \\ \exp\{\theta_{h,m}\}, & \text{otherwise} \end{cases}$$

To account for the dependencies $(0, m)$ headed by the root-symbol, we define a vector of root-selection scores $\mathbf{r}(\boldsymbol{\theta}) \in \mathbb{R}^n$, for $m = 1 \ldots n$:

$$r_m(\boldsymbol{\theta}) \;=\; \exp\{\theta_{0,m}\}$$

Let the weight of a dependency structure $y \in \mathcal{T}^s_{np}(\mathbf{x})$ be defined as:

$$\psi(y; \boldsymbol{\theta}) \;=\; r_{\text{root}(y)}(\boldsymbol{\theta}) \prod_{(h,m) \in y \,:\, h \neq 0} A_{h,m}(\boldsymbol{\theta})$$

Here, $\text{root}(y) = m : (0, m) \in y$ is the child of the root-symbol; there is exactly one such child, since $y \in \mathcal{T}^s_{np}(\mathbf{x})$. Eq. 2 and 3 can be rephrased as:

$$P(y \,|\, \mathbf{x}; \boldsymbol{\theta}) \;=\; \frac{\psi(y; \boldsymbol{\theta})}{Z(\mathbf{x}; \boldsymbol{\theta})} \qquad (4)$$

$$Z(\mathbf{x}; \boldsymbol{\theta}) \;=\; \sum_{y \in \mathcal{T}^s_{np}(\mathbf{x})} \psi(y; \boldsymbol{\theta}) \qquad (5)$$

In the remainder of this section, we drop the notational dependence on $\mathbf{x}$ for brevity.

The original Matrix-Tree Theorem addressed the problem of counting the number of undirected spanning trees in an undirected graph. For the models we study here, we require a sum of *weighted* and *directed* spanning trees. Tutte (1984) extended the Matrix-Tree Theorem to this case. We briefly summarize his method below.

First, define the Laplacian matrix $L(\boldsymbol{\theta}) \in \mathbb{R}^{n \times n}$ of G, for $h, m = 1 \ldots n$:

$$L_{h,m}(\boldsymbol{\theta}) \;=\; \begin{cases} \sum_{h'=1}^{n} A_{h',m}(\boldsymbol{\theta}) & \text{if } h = m \\ -A_{h,m}(\boldsymbol{\theta}) & \text{otherwise} \end{cases}$$

Second, for a matrix X, let $X^{(h,m)}$ be the *minor* of X with respect to row h and column m; i.e., the determinant of the matrix formed by deleting row h and column m from X. Finally, define the weight of any directed spanning tree of G to be the product of the weights $A_{h,m}(\boldsymbol{\theta})$ for the edges in that tree.

Theorem 1 *(Tutte, 1984, p. 140). Let $L(\boldsymbol{\theta})$ be the Laplacian matrix of G. Then $L^{(m,m)}(\boldsymbol{\theta})$ is equal to the sum of the weights of all directed spanning trees of G which are rooted at m. Furthermore, the minors vary only in sign when traversing the columns of the Laplacian (Tutte, 1984, p. 150):*

$$\forall h, m\colon\; (-1)^{h+m} L^{(h,m)}(\boldsymbol{\theta}) \;=\; L^{(m,m)}(\boldsymbol{\theta}) \qquad (6)$$

3.1 Partition functions via matrix determinants

From Theorem 1, it directly follows that

$$L^{(m,m)}(\boldsymbol{\theta}) \;=\; \sum_{y \in \mathcal{U}(m)} \;\prod_{(h,m) \in y \,:\, h \neq 0} A_{h,m}(\boldsymbol{\theta})$$

where $\mathcal{U}(m) = \{y \in \mathcal{T}^s_{np} : \text{root}(y) = m\}$. A naïve method for computing the partition function is therefore to evaluate

$$Z(\boldsymbol{\theta}) \;=\; \sum_{m=1}^{n} r_m(\boldsymbol{\theta}) L^{(m,m)}(\boldsymbol{\theta})$$

The above would require calculating n determinants, resulting in $O(n^4)$ complexity. However, as we show below $Z(\boldsymbol{\theta})$ may be obtained in $O(n^3)$ time using a single determinant evaluation.

Define a new matrix $\hat{L}(\boldsymbol{\theta})$ to be $L(\boldsymbol{\theta})$ with the first row replaced by the root-selection scores:

$$\hat{L}_{h,m}(\boldsymbol{\theta}) \;=\; \begin{cases} r_m(\boldsymbol{\theta}) & h = 1 \\ L_{h,m}(\boldsymbol{\theta}) & h > 1 \end{cases}$$

This matrix allows direct computation of the partition function, as the following proposition shows.

Proposition 1 *The partition function in Eq. 5 is given by $Z(\boldsymbol{\theta}) = |\hat{L}(\boldsymbol{\theta})|$.*

Proof: *Consider the row expansion of $|\hat{L}(\boldsymbol{\theta})|$ with respect to row 1:*

$$\begin{aligned} |\hat{L}(\boldsymbol{\theta})| &= \sum_{m=1}^{n} (-1)^{1+m} \hat{L}_{1,m}(\boldsymbol{\theta}) \hat{L}^{(1,m)}(\boldsymbol{\theta}) \\ &= \sum_{m=1}^{n} (-1)^{1+m} r_m(\boldsymbol{\theta}) L^{(1,m)}(\boldsymbol{\theta}) \\ &= \sum_{m=1}^{n} r_m(\boldsymbol{\theta}) L^{(m,m)}(\boldsymbol{\theta}) \;=\; Z(\boldsymbol{\theta}) \end{aligned}$$

The second line follows from the construction of $\hat{L}(\boldsymbol{\theta})$, and the third line follows from Eq. 6. ■

3.2 Marginals via matrix inversion

The marginals we require are given by

$$\mu_{h,m}(\boldsymbol{\theta}) \;=\; \frac{1}{Z(\boldsymbol{\theta})} \sum_{y \in \mathcal{T}^s_{np} :\, (h,m) \in y} \psi(y; \boldsymbol{\theta})$$

To calculate these marginals efficiently for all values of (h, m) we use a well-known identity relating the log partition-function to marginals

$$\mu_{h,m}(\boldsymbol{\theta}) \;=\; \frac{\partial \log Z(\boldsymbol{\theta})}{\partial \theta_{h,m}}$$

Since the partition function in this case has a closed-form expression (i.e., the determinant of a matrix constructed from $\boldsymbol{\theta}$), the marginals can also obtained in closed form. Using the chain rule, the derivative of the log partition-function in Proposition 1 is

$$\begin{aligned} \mu_{h,m}(\boldsymbol{\theta}) &= \frac{\partial \log |\hat{L}(\boldsymbol{\theta})|}{\partial \theta_{h,m}} \\ &= \sum_{h'=1}^{n} \sum_{m'=1}^{n} \frac{\partial \log |\hat{L}(\boldsymbol{\theta})|}{\partial \hat{L}_{h',m'}(\boldsymbol{\theta})} \frac{\partial \hat{L}_{h',m'}(\boldsymbol{\theta})}{\partial \theta_{h,m}} \end{aligned}$$

To perform the derivative, we use the identity

$$\frac{\partial \log |X|}{\partial X} \;=\; \left(X^{-1}\right)^T$$

and the fact that $\partial \hat{L}_{h',m'}(\boldsymbol{\theta}) / \partial \theta_{h,m}$ is nonzero for only a few h', m'. Specifically, when $h = 0$, the marginals are given by

$$\mu_{0,m}(\boldsymbol{\theta}) \;=\; r_m(\boldsymbol{\theta}) \left[\hat{L}^{-1}(\boldsymbol{\theta})\right]_{m,1}$$

and for $h > 0$, the marginals are given by

$$\begin{aligned} \mu_{h,m}(\boldsymbol{\theta}) \;=\; & (1 - \delta_{1,m}) A_{h,m}(\boldsymbol{\theta}) \left[\hat{L}^{-1}(\boldsymbol{\theta})\right]_{m,m} - \\ & (1 - \delta_{h,1}) A_{h,m}(\boldsymbol{\theta}) \left[\hat{L}^{-1}(\boldsymbol{\theta})\right]_{m,h} \end{aligned}$$

where $\delta_{h,m}$ is the Kronecker delta. Thus, the complexity of evaluating *all* the relevant marginals is dominated by the matrix inversion, and the total complexity is therefore $O(n^3)$.

3.3 Multiple Roots

In the case of multiple roots, we can still compute the partition function and marginals efficiently. In fact, the derivation of this case is simpler than for single-root structures. Create an extended graph G' which augments G with a dummy root node that has edges pointing to all of the existing nodes, weighted by the appropriate root-selection scores. Note that there is a bijection between directed spanning trees of G' rooted at the dummy root and multi-root structures $y \in \mathcal{T}^m_{np}(\mathbf{x})$. Thus, Theorem 1 can be used to compute the partition function directly: construct a Laplacian matrix $L(\boldsymbol{\theta})$ for G' and compute the minor $L^{(0,0)}(\boldsymbol{\theta})$. Since this minor is also a determinant, the marginals can be obtained analogously to the single-root case. More concretely, this technique corresponds to defining the matrix $\hat{L}(\boldsymbol{\theta})$ as

$$\hat{L}(\boldsymbol{\theta}) \;=\; L(\boldsymbol{\theta}) + \mathrm{diag}(\mathbf{r}(\boldsymbol{\theta}))$$

where $\mathrm{diag}(\mathbf{v})$ is the diagonal matrix with the vector $\mathbf{v}$ on its diagonal.

3.4 Labeled Trees

The techniques above extend easily to the case where dependencies are labeled. For a model with L different labels, it suffices to define the edge and root scores as $A_{h,m}(\boldsymbol{\theta}) = \sum_{\ell=1}^{L} \exp\{\theta_{h,m,\ell}\}$ and $r_m(\boldsymbol{\theta}) = \sum_{\ell=1}^{L} \exp\{\theta_{0,m,\ell}\}$. The partition function over labeled trees is obtained by operating on these values as described previously, and the marginals are given by an application of the chain rule. Both inference problems are solvable in $O(n^3 + Ln^2)$ time.

4 Training Algorithms

This section describes two methods for parameter estimation that rely explicitly on the computation of the partition function and marginals.

4.1 Log-Linear Estimation

In conditional log-linear models (Johnson et al., 1999; Lafferty et al., 2001), a distribution over parse trees for a sentence $\mathbf{x}$ is defined as follows:

$$P(y \,|\, \mathbf{x}; \mathbf{w}) \;=\; \frac{\exp\left\{\sum_{(h,m) \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)\right\}}{Z(\mathbf{x}; \mathbf{w})} \qquad (7)$$

where $Z(\mathbf{x}; \mathbf{w})$ is the partition function, a sum over $\mathcal{T}^s_p(\mathbf{x})$, $\mathcal{T}^s_{np}(\mathbf{x})$, $\mathcal{T}^m_p(\mathbf{x})$ or $\mathcal{T}^m_{np}(\mathbf{x})$.

We train the model using the approach described by Sha and Pereira (2003). Assume that we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. The optimal parameters are taken to be $\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}} L(\mathbf{w})$ where

$$L(\mathbf{w}) \;=\; -C\sum_{i=1}^{N} \log P(y_i \,|\, \mathbf{x}_i; \mathbf{w}) + \frac{1}{2}||\mathbf{w}||^2$$

The parameter $C > 0$ is a constant dictating the level of regularization in the model.

Since $L(\mathbf{w})$ is a convex function, gradient descent methods can be used to search for the global minimum. Such methods typically involve repeated computation of the loss $L(\mathbf{w})$ and gradient $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$, requiring efficient implementations of both functions. Note that the log-probability of a parse is

$$\log P(y \,|\, \mathbf{x}; \mathbf{w}) = \sum_{(h,m)\in y} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, h, m) - \log Z(\mathbf{x}; \mathbf{w})$$

so that the main issue in calculating the loss function $L(\mathbf{w})$ is the evaluation of the partition functions $Z(\mathbf{x}_i; \mathbf{w})$. The gradient of the loss is given by

$$\begin{aligned}\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \;&=\; \mathbf{w} - C\sum_{i=1}^{N} \sum_{(h,m)\in y_i} \mathbf{f}(\mathbf{x}_i, h, m) \\ &+ C\sum_{i=1}^{N} \sum_{(h,m)\in \mathcal{D}(\mathbf{x}_i)} \mu_{h,m}(\mathbf{x}_i; \mathbf{w})\mathbf{f}(\mathbf{x}_i, h, m)\end{aligned}$$

where

$$\mu_{h,m}(\mathbf{x}; \mathbf{w}) \;=\; \sum_{y\in\mathcal{T}(\mathbf{x})\,:\,(h,m)\in y} P(y \,|\, \mathbf{x}; \mathbf{w})$$

is the marginal probability of a dependency (h, m). Thus, the main issue in the evaluation of the gradient is the computation of the marginals $\mu_{h,m}(\mathbf{x}_i; \mathbf{w})$.

Note that Eq. 7 forms a special case of the log-linear distribution defined in Eq. 2 in Section 2.2. If we set $\theta_{h,m} = \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, h, m)$ then we have $P(y \,|\, \mathbf{x}; \mathbf{w}) = P(y \,|\, \mathbf{x}; \boldsymbol{\theta})$, $Z(\mathbf{x}; \mathbf{w}) = Z(\mathbf{x}; \boldsymbol{\theta})$, and $\mu_{h,m}(\mathbf{x}; \mathbf{w}) = \mu_{h,m}(\mathbf{x}; \boldsymbol{\theta})$. Thus in the projective case the inside-outside algorithm can be used to calculate the partition function and marginals, thereby enabling training of a log-linear model; in the non-projective case the algorithms in Section 3 can be used for this purpose.

4.2 Max-Margin Estimation

The second learning algorithm we consider is the large-margin approach for structured prediction (Taskar et al., 2004a; Taskar et al., 2004b). Learning in this framework again involves minimization of a convex function $L(\mathbf{w})$. Let the *margin* for parse tree y on the i'th training example be defined as

$$m_{i,y}(\mathbf{w}) = \sum_{(h,m)\in y_i} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}_i, h, m) - \sum_{(h,m)\in y} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}_i, h, m)$$

The loss function is then defined as

$$L(\mathbf{w}) = C\sum_{i=1}^{N} \max_{y\in\mathcal{T}(\mathbf{x}_i)} \left(E_{i,y} - m_{i,y}(\mathbf{w})\right) + \frac{1}{2}||\mathbf{w}||^2$$

where $E_{i,y}$ is a measure of the loss—or number of errors—for parse y on the i'th training sentence. In this paper we take $E_{i,y}$ to be the number of incorrect dependencies in the parse tree y when compared to the gold-standard parse tree y_i.

The definition of $L(\mathbf{w})$ makes use of the expression $\max_{y\in\mathcal{T}(\mathbf{x}_i)} (E_{i,y} - m_{i,y}(\mathbf{w}))$ for the i'th training example, which is commonly referred to as the *hinge loss*. Note that $E_{i,y_i} = 0$, and also that $m_{i,y_i}(\mathbf{w}) = 0$, so that the hinge loss is always non-negative. In addition, the hinge loss is 0 if and only if $m_{i,y}(\mathbf{w}) \geq E_{i,y}$ for all $y \in \mathcal{T}(\mathbf{x}_i)$. Thus the hinge loss directly penalizes margins $m_{i,y}(\mathbf{w})$ which are less than their corresponding losses $E_{i,y}$.

Figure 2 shows an algorithm for minimizing $L(\mathbf{w})$ that is based on the exponentiated-gradient algorithm for large-margin optimization described by Bartlett et al. (2004). The algorithm maintains a set of weights $\theta_{i,h,m}$ for $i = 1 \ldots N, (h, m) \in \mathcal{D}(\mathbf{x}_i)$, which are updated example-by-example. The algorithm relies on the repeated computation of marginal values $\mu_{i,h,m}$, which are defined as follows:[1]

$$\begin{aligned}\mu_{i,h,m} \;&=\; \sum_{y\in\mathcal{T}(\mathbf{x}_i)\,:\,(h,m)\in y} P(y \,|\, \mathbf{x}_i) \qquad (8) \\ P(y \,|\, \mathbf{x}_i) \;&=\; \frac{\exp\left\{\sum_{(h,m)\in y} \theta_{i,h,m}\right\}}{\sum_{y'\in\mathcal{T}(\mathbf{x}_i)} \exp\left\{\sum_{(h,m)\in y'} \theta_{i,h,m}\right\}}\end{aligned}$$

A similar definition is used to derive marginal values $\mu'_{i,h,m}$ from the values $\theta'_{i,h,m}$. Computation of the μ and μ' values is again inference of the form described in Problem 3 in Section 2.2, and can be

[1] Bartlett et al. (2004) write $P(y \,|\, \mathbf{x}_i)$ as $\alpha_{i,y}$. The $\alpha_{i,y}$ variables are dual variables that appear in the dual objective function, i.e., the convex dual of $L(\mathbf{w})$. Analysis of the algorithm shows that as the $\theta_{i,h,m}$ variables are updated, the dual variables converge to the optimal point of the dual objective, and the parameters $\mathbf{w}$ converge to the minimum of $L(\mathbf{w})$.

Inputs: Training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$.

Parameters: Regularization constant C, starting point β, number of passes over training set T.

Data Structures: Real values $\theta_{i,h,m}$ and $l_{i,h,m}$ for $i = 1 \ldots N, (h,m) \in \mathcal{D}(\mathbf{x}_i)$. Learning rate η.

Initialization: Set learning rate $\eta = \frac{1}{C}$. Set $\theta_{i,h,m} = \beta$ for $(h,m) \in y_i$, and $\theta_{i,h,m} = 0$ for $(h,m) \notin y_i$. Set $l_{i,h,m} = 0$ for $(h,m) \in y_i$, and $l_{i,h,m} = 1$ for $(h,m) \notin y_i$. Calculate initial parameters as

$$\mathbf{w} = C \sum_i \sum_{(h,m) \in \mathcal{D}(\mathbf{x}_i)} \delta_{i,h,m} \mathbf{f}(\mathbf{x}_i, h, m)$$

where $\delta_{i,h,m} = (1 - l_{i,h,m} - \mu_{i,h,m})$ and the $\mu_{i,h,m}$ values are calculated from the $\theta_{i,h,m}$ values as described in Eq. 8.

Algorithm: Repeat T passes over the training set, where each pass is as follows:

```
Set $obj = 0$

For $i = 1 \ldots N$
  • For all $(h,m) \in \mathcal{D}(\mathbf{x}_i)$:
    $\theta'_{i,h,m} = \theta_{i,h,m} + \eta C \left( l_{i,h,m} + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, h, m) \right)$

    • For example $i$, calculate marginals $\mu_{i,h,m}$
    from $\theta_{i,h,m}$ values, and marginals $\mu'_{i,h,m}$
    from $\theta'_{i,h,m}$ values (see Eq. 8).

  • Update the parameters:
    $\mathbf{w} = \mathbf{w} + C \sum_{(h,m) \in \mathcal{D}(\mathbf{x}_i)} \delta_{i,h,m} \mathbf{f}(\mathbf{x}_i, h, m)$
    where $\delta_{i,h,m} = \mu_{i,h,m} - \mu'_{i,h,m}$,

  • For all $(h,m) \in \mathcal{D}(\mathbf{x}_i)$, set $\theta_{i,h,m} = \theta'_{i,h,m}$

  • Set $obj = obj + C \sum_{(h,m) \in \mathcal{D}(\mathbf{x}_i)} l_{i,h,m} \mu'_{i,h,m}$

Set $obj = obj - \frac{||\mathbf{w}||^2}{2}$. If $obj$ has decreased
compared to last iteration, set $\eta = \frac{\eta}{2}$.
```

Output: Parameter values $\mathbf{w}$.

Figure 2: The EG Algorithm for Max-Margin Estimation. The learning rate η is halved each time the dual objective function (see (Bartlett et al., 2004)) fails to increase. In our experiments we chose $\beta = 9$, which was found to work well during development of the algorithm.

achieved using the inside-outside algorithm for projective structures, and the algorithms described in Section 3 for non-projective structures.

5 Related Work

Global log-linear training has been used in the context of PCFG parsing (Johnson, 2001). Riezler et al. (2004) explore a similar application of log-linear models to LFG parsing. Max-margin learning has been applied to PCFG parsing by Taskar et al. (2004b). They show that this problem has a QP dual of polynomial size, where the dual variables correspond to marginal probabilities of CFG rules. A similar QP dual may be obtained for max-margin projective dependency parsing. However, for non-projective parsing, the dual QP would require an exponential number of constraints on the dependency marginals (Chopra, 1989). Nevertheless, alternative optimization methods like that of Tsochantaridis et al. (2004), or the EG method presented here, can still be applied.

The majority of previous work on dependency parsing has focused on local (i.e., classification of individual edges) discriminative training methods (Yamada and Matsumoto, 2003; Nivre et al., 2004; Y. Cheng, 2005). Non-local (i.e., classification of entire trees) training methods were used by McDonald et al. (2005a), who employed online learning.

Dependency parsing accuracy can be improved by allowing second-order features, which consider more than one dependency simultaneously. McDonald and Pereira (2006) define a second-order dependency parsing model in which interactions between adjacent siblings are allowed, and Carreras (2007) defines a second-order model that allows grandparent and sibling interactions. Both authors give polytime algorithms for exact projective parsing. By adapting the inside-outside algorithm to these models, partition functions and marginals can be computed for second-order projective structures, allowing log-linear and max-margin training to be applied via the framework developed in this paper. For higher-order non-projective parsing, however, computational complexity results (McDonald and Pereira, 2006; McDonald and Satta, 2007) indicate that exact solutions to the three inference problems of Section 2.2 will be intractable. Exploration of approximate second-order non-projective inference is a natural avenue for future research.

Two other groups of authors have independently and simultaneously proposed adaptations of the Matrix-Tree Theorem for structured inference on directed spanning trees (McDonald and Satta, 2007; Smith and Smith, 2007). There are some algorithmic differences between these papers and ours. First, we define both multi-root and single-root algorithms, whereas the other papers only consider multi-root parsing. This distinction can be important as one often expects a dependency structure to have exactly one child attached to the root-symbol, as is the case in a single-root structure. Second, McDonald and Satta (2007) propose an $O(n^5)$ algorithm for computing the marginals, as opposed to the $O(n^3)$ matrix-inversion approach used by Smith and Smith (2007) and ourselves.

In addition to the algorithmic differences, both groups of authors consider applications of the Matrix-Tree Theorem which we have not discussed. For example, both papers propose minimum-risk decoding, and McDonald and Satta (2007) discuss unsupervised learning and language modeling, while Smith and Smith (2007) define hidden-variable models based on spanning trees.

In this paper we used EG training methods only for max-margin models (Bartlett et al., 2004). However, Globerson et al. (2007) have recently shown how EG updates can be applied to efficient training of log-linear models.

6 Experiments on Dependency Parsing

In this section, we present experimental results applying our inference algorithms for dependency parsing models. Our primary purpose is to establish comparisons along two relevant dimensions: projective training vs. non-projective training, and marginal-based training algorithms vs. the averaged perceptron. The feature representation and other relevant dimensions are kept fixed in the experiments.

6.1 Data Sets and Features

We used data from the CoNLL-X shared task on multilingual dependency parsing (Buchholz and Marsi, 2006). In our experiments, we used a subset consisting of six languages; Table 1 gives details of the data sets used.[2] For each language we created a validation set that was a subset of the CoNLL-X training set for that language. The remainder of each training set was used to train the models for the different languages. The validation sets were used to tune the meta-parameters (e.g., the value of the regularization constant C) of the different training algorithms. We used the official test sets and evaluation script from the CoNLL-X task. All of the results that we report are for unlabeled dependency parsing.[3]

language	%cd	train	val.	test
Arabic	0.34	49,064	5,315	5,373
Dutch	4.93	178,861	16,208	5,585
Japanese	0.70	141,966	9,495	5,711
Slovene	1.59	22,949	5,801	6,390
Spanish	0.06	78,310	11,024	5,694
Turkish	1.26	51,827	5,683	7,547

Table 1: Information for the languages in our experiments. The 2nd column (%cd) is the percentage of crossing dependencies in the training and validation sets. The last three columns report the size in tokens of the training, validation and test sets.

The non-projective models were trained on the CoNLL-X data in its original form. Since the projective models assume that the dependencies in the data are non-crossing, we created a second training set for each language where non-projective dependency structures were automatically transformed into projective structures. All projective models were trained on these new training sets.[4] Our feature space is based on that of McDonald et al. (2005a).[5]

6.2 Results

We performed experiments using three training algorithms: the averaged perceptron (Collins, 2002), log-linear training (via conjugate gradient descent), and max-margin training (via the EG algorithm). Each of these algorithms was trained using projective and non-projective methods, yielding six training settings per language. The different training algorithms have various meta-parameters, which we optimized on the validation set for each language/training-setting combination. The

[2] Our subset includes the two languages with the lowest accuracy in the CoNLL-X evaluations (Turkish and Arabic), the language with the highest accuracy (Japanese), the most non-projective language (Dutch), a moderately non-projective language (Slovene), and a highly projective language (Spanish). All languages but Spanish have multi-root parses in their data. We are grateful to the providers of the treebanks that constituted the data of our experiments (Hajič et al., 2004; van der Beek et al., 2002; Kawata and Bartels, 2000; Džeroski et al., 2006; Civit and Martí, 2002; Oflazer et al., 2003).

[3] Our algorithms also support labeled parsing (see Section 3.4). Initial experiments with labeled models showed the same trend that we report here for unlabeled parsing, so for simplicity we conducted extensive experiments only for unlabeled parsing.

[4] The transformations were performed by running the projective parser with score +1 on correct dependencies and -1 otherwise: the resulting trees are guaranteed to be projective and to have a minimum loss with respect to the correct tree. Note that only the training sets were transformed.

[5] It should be noted that McDonald et al. (2006) use a richer feature set that is incomparable to our features.

	Perceptron		Max-Margin		Log-Linear	
	p	np	p	np	p	np
Ara	71.74	71.84	71.74	72.99	73.11	**73.67**
Dut	77.17	78.83	76.53	**79.69**	76.23	79.55
Jap	91.90	91.78	92.10	**92.18**	91.68	91.49
Slo	78.02	78.66	79.78	**80.10**	78.24	79.66
Spa	81.19	80.02	81.71	**81.93**	81.75	81.57
Tur	71.22	71.70	**72.83**	72.02	72.26	72.62

Table 2: Test data results. The p and np columns show results with projective and non-projective training respectively.

	Ara	Dut	Jap	Slo	Spa	Tur	AV
P	71.74	78.83	91.78	78.66	81.19	71.70	79.05
E	72.99	**79.69**	**92.18**	**80.10**	**81.93**	72.02	**79.82**
L	**73.67**	79.55	91.49	79.66	81.57	**72.26**	79.71

Table 3: Results for the three training algorithms on the different languages (P = perceptron, E = EG, L = log-linear models). AV is an average across the results for the different languages.

averaged perceptron has a single meta-parameter, namely the number of iterations over the training set. The log-linear models have two meta-parameters: the regularization constant C and the number of gradient steps T taken by the conjugate-gradient optimizer. The EG approach also has two meta-parameters: the regularization constant C and the number of iterations, T.[6] For models trained using non-projective algorithms, both projective and non-projective parsing was tested on the validation set, and the highest scoring of these two approaches was then used to decode test data sentences.

Table 2 reports test results for the six training scenarios. These results show that for Dutch, which is the language in our data that has the highest number of crossing dependencies, non-projective training gives significant gains over projective training for all three training methods. For the other languages, non-projective training gives similar or even improved performance over projective training.

Table 3 gives an additional set of results, which were calculated as follows. For each of the three training methods, we used the validation set results to choose between projective and non-projective training. This allows us to make a direct comparison of the three training algorithms. Table 3 shows the results of this comparison.[7] The results show that log-linear and max-margin models both give a higher average accuracy than the perceptron. For some languages (e.g., Japanese), the differences from the perceptron are small; however for other languages (e.g., Arabic, Dutch or Slovene) the improvements seen are quite substantial.

7 Conclusions

This paper describes inference algorithms for spanning-tree distributions, focusing on the fundamental problems of computing partition functions and marginals. Although we concentrate on log-linear and max-margin estimation, the inference algorithms we present can serve as black-boxes in many other statistical modeling techniques.

Our experiments suggest that marginal-based training produces more accurate models than perceptron learning. Notably, this is the first large-scale application of the EG algorithm, and shows that it is a promising approach for structured learning.

In line with McDonald et al. (2005b), we confirm that spanning-tree models are well-suited to dependency parsing, especially for highly non-projective languages such as Dutch. Moreover, spanning-tree models should be useful for a variety of other problems involving structured data.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments. In addition, the authors gratefully acknowledge the following sources of support. Terry Koo was funded by a grant from the NSF (DMS-0434222) and a grant from NTT, Agmt. Dtd. 6/21/1998. Amir Globerson was supported by a fellowship from the Rothschild Foundation - Yad Hanadiv. Xavier Carreras was supported by the Catalan Ministry of Innovation, Universities and Enterprise, and a grant from NTT, Agmt. Dtd. 6/21/1998. Michael Collins was funded by NSF grants 0347631 and DMS-0434222.

[6] We trained the perceptron for 100 iterations, and chose the iteration which led to the best score on the validation set. Note that in all of our experiments, the best perceptron results were actually obtained with 30 or fewer iterations. For the log-linear and EG algorithms we tested a number of values for C, and for each value of C ran 100 gradient steps or EG iterations, finally choosing the best combination of C and T found in validation.

[7] We ran the sign test at the sentence level to measure the statistical significance of the results aggregated across the six languages. Out of 2,472 sentences total, log-linear models gave improved parses over the perceptron on 448 sentences, and worse parses on 343 sentences. The max-margin method gave improved/worse parses for 500/383 sentences. Both results are significant with $p \leq 0.001$.

References

J. Baker. 1979. Trainable grammars for speech recognition. In *97th meeting of the Acoustical Society of America*.

P. Bartlett, M. Collins, B. Taskar, and D. McAllester. 2004. Exponentiated gradient algorithms for large–margin structured classification. In *NIPS*.

L.E. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41:164–171.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. CoNLL-X*.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. EMNLP-CoNLL*.

S. Chopra. 1989. On the spanning tree polyhedron. *Oper. Res. Lett.*, pages 25–29.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Civit and M^{a} A. Martí. 2002. Design principles for a Spanish treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.

A. Globerson, T. Koo, X. Carreras, and M. Collins. 2007. Exponentiated gradient algorithms for log-linear structured prediction. In *Proc. ICML*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic unification-based grammars. In *Proc. ACL*.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. ACL*.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditonal random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. EACL*.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. IWPT*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT-EMNLP*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency parsing with a two-stage discriminative parser. In *Proc. CoNLL-X*.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. CoNLL*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, chapter 15. Kluwer Academic Publishers.

M.A. Paskin. 2001. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical Report UCB/CSD-01-1148, University of California, Berkeley.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd edition)*. Morgan Kaufmann Publishers.

S. Riezler, R. Kaplan, T. King, J. Maxwell, A. Vasserman, and R. Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proc. HLT-NAACL*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL*.

N.A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. ACL*.

D.A. Smith and N.A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proc. EMNLP-CoNLL*.

B. Taskar, C. Guestrin, and D. Koller. 2004a. Max-margin markov networks. In *NIPS*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004b. Max-margin parsing. In *Proc. EMNLP*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*.

W. Tutte. 1984. *Graph Theory*. Addison-Wesley.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Y. Matsumoto Y. Cheng, M. Asahara. 2005. Machine learning-based dependency analyzer for chinese. In *Proc. ICCC*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.

Using Foreign Inclusion Detection to Improve Parsing Performance

Beatrice Alex, Amit Dubey and Frank Keller
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW, UK
{balex,adubey,keller}@inf.ed.ac.uk

Abstract

Inclusions from other languages can be a significant source of errors for monolingual parsers. We show this for English inclusions, which are sufficiently frequent to present a problem when parsing German. We describe an annotation-free approach for accurately detecting such inclusions, and develop two methods for interfacing this approach with a state-of-the-art parser for German. An evaluation on the TIGER corpus shows that our inclusion entity model achieves a performance gain of 4.3 points in F-score over a baseline of no inclusion detection, and even outperforms a parser with access to gold standard part-of-speech tags.

1 Introduction

The status of English as a global language means that English words and phrases are frequently borrowed by other languages, especially in domains such as science and technology, commerce, advertising, and current affairs. This is an instance of *language mixing*, whereby inclusions from other languages appear in an otherwise monolingual text. While the processing of foreign inclusions has received some attention in the text-to-speech (TTS) literature (see Section 2), the natural language processing (NLP) community has paid little attention both to the problem of inclusion detection, and to potential applications thereof. Also the extent to which inclusions pose a problem to existing NLP methods has not been investigated.

In this paper, we address this challenge. We focus on English inclusions in German text. Anglicisms and other borrowings from English form by far the most frequent foreign inclusions in German. In specific domains, up to 6.4% of the tokens of a German text can be English inclusions. Even in regular newspaper text as used for many NLP applications, English inclusions can be found in up to 7.4% of all sentences (see Section 3 for both figures).

Virtually all existing NLP algorithms assume that the input is monolingual, and does not contain foreign inclusions. It is possible that this is a safe assumption, and inclusions can be dealt with accurately by existing methods, without resorting to specialized mechanisms. The alternative hypothesis, however, seems more plausible: foreign inclusions pose a problem for existing approaches, and sentences containing them are processed less accurately. A parser, for example, is likely to have problems with inclusions – most of the time, they are unknown words, and as they originate from another language, standard methods for unknown words guessing (suffix stripping, etc.) are unlikely to be successful. Furthermore, the fact that inclusions are often multiword expressions (e.g., named entities) means that simply part-of-speech (POS) tagging them accurately is not sufficient: if the parser posits a phrase boundary within an inclusion this is likely to severely decrease parsing accuracy.

In this paper, we focus on the impact of English inclusions on the parsing of German text. We describe an annotation-free method that accurately recognizes English inclusions, and demonstrate that inclusion detection improves the performance of a state-of-the-art parser for German. We show that the way of interfacing the inclusion detection and the parser is crucial, and propose a method for modifying the underlying probabilistic grammar in order to

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 151–160, Prague, June 2007.

enable the parser to process inclusions accurately.

This paper is organized as follows. We review related work in Section 2, and present the English inclusion classifier in Section 3. Section 4 describes our results on interfacing inclusion detection with parsing, and Section 5 presents an error analysis. Discussion and conclusion follow in Section 6.

2 Related Work

Previous work on inclusion detection exists in the TTS literature. Here, the aim is to design a system that recognizes foreign inclusions on the word and sentence level and functions at the front-end to a polyglot TTS synthesizer. Pfister and Romsdorfer (2003) propose morpho-syntactic analysis combined with lexicon lookup to identify foreign words in mixed-lingual text. While they state that their system is precise at detecting the language of tokens and determining the sentence structure, it is not evaluated on real mixed-lingual text. A further approach to inclusion detection is that of Marcadet et. al (2005). They present experiments with a dictionary-driven transformation-based learning method and a corpus-based n-gram approach and show that a combination of both methods yields the best results. Evaluated on three mixed-lingual test sets in different languages, the combined approach yields word-based language identification error rates (i.e. the percentage of tokens for which the language is identified incorrectly) of 0.78% on the French data, 1.33% on the German data and 0.84% on the Spanish data. Consisting of 50 sentences or less for each language, their test sets are very small and appear to be selected specifically for evaluation purposes. It would therefore be interesting to determine the system's performance on random and unseen data and examine how it scales up to larger data sets.

Andersen (2005), noting the importance of recognizing anglicisms to lexicographers, tests algorithms based on lexicon lookup, character n-grams and regular expressions and a combination thereof to automatically extract anglicisms in Norwegian text. On a 10,000 word subset of the neologism archive (Wangensteen, 2002), the best method of combining character n-grams and regular expression matching yields an accuracy of 96.32% and an F-score of 59.4 (P = 75.8%, R = 48.8%). This result is unsurprisingly low as no differentiation is made between full-word anglicisms and tokens with mixed-lingual morphemes in the gold standard.

In the context of parsing, Forst and Kaplan (2006) have observed that the failure to properly deal with foreign inclusions is detrimental to a parser's accuracy. However, they do not substantiate this claim using numeric results.

3 English Inclusion Detection

Previous work reported by Alex (2006; 2005) has focused on devising a classifier that detects anglicisms and other English inclusions in text written in other languages, namely German and French. This inclusion classifier is based on a lexicon and search engine lookup as well as a post-processing step.

The lexicon lookup is performed for tokens tagged as noun (*NN*), named entity (*NE*), foreign material (*FM*) or adjective (*ADJA/ADJD*) using the German and English CELEX lexicons. Tokens only found in the English lexicon are classified as English. Tokens found in neither lexicon are passed to the search engine module. Tokens found in both databases are classified by the post-processing module. The search engine module performs language classification based on the maximum normalised score of the number of hits returned for two searches per token, one for each language (Alex, 2005). This score is determined by weighting the number of hits, i.e. the "absolute frequency" by the estimated size of the accessible Web corpus for that language (Alex, 2006). Finally, the rule-based post-processing module classifies single-character tokens and resolves language classification ambiguities for interlingual homographs, English function words, names of currencies and units of measurement. A further post-processing step relates language information between abbreviations or acronyms and their definitions in combination with an abbreviation extraction algorithm (Schwartz and Hearst, 2003). Finally, a set of rules disambiguates English inclusions from person names (Alex, 2006).

For German, the classifier has been evaluated on test sets in three different domains: newspaper articles, selected from the Frankfurter Allgemeine Zeitung, on internet and telecoms, space travel and European Union related topics. Table 1 presents an

Domain	EI tokens	EI types	EI TTR	Accuracy	Precision	Recall	F
Internet	6.4%	5.9%	0.25	98.13%	91.58%	78.92%	84.78
Space	2.8%	3.5%	0.33	98.97%	84.02%	85.31%	84.66
EU	1.1%	2.1%	0.50	99.65%	82.16%	87.36%	84.68

Table 1: English inclusion (EI) token and type statistics, EI type-token-ratios (TTR) as well as accuracy, precision, recall and F-scores for the unseen German test sets.

overview of the percentages of English inclusion tokens and types within the gold standard annotation of each test set, and illustrates how well the English inclusion classifier is able to detect them in terms of F-score. The figures show that the frequency of English inclusions varies considerably depending on the domain but that the classifier is able to detect them equally well with an F-score approaching 85 for each domain.

The recognition of English inclusions bears similarity to classification tasks such as named entity recognition, for which various machine learning (ML) techniques have proved successful. In order to compare the performance of the English inclusion classifier against a trained ML classifier, we pooled the annotated English inclusion evaluation data for all three domains. As the English inclusion classifier does not rely on annotated data, it can be tested and evaluated once for the entire corpus. The ML classifier used for this experiment is a conditional Markov model tagger which is designed for, and proved successful in, named entity recognition in newspaper and biomedical text (Klein et al., 2003; Finkel et al., 2005). It can be trained to perform similar information extraction tasks such as English inclusion detection. To determine the tagger's performance over the entire set and to investigate the effect of the amount of annotated training data available, a 10-fold cross-validation test was conducted whereby increasing sub-parts of the training data are provided when testing on each fold. The resulting learning curves in Figure 1 show that the English inclusion classifier has an advantage over the supervised ML approach, despite the fact the latter requires expensive hand-annotated data. A large training set of 80,000 tokens is required to yield a performance that approximates that of our annotation-free inclusion classifier. This system has been shown to perform similarly well on unseen texts in different domains, plus it is easily extendable to a new language (Alex, 2006).

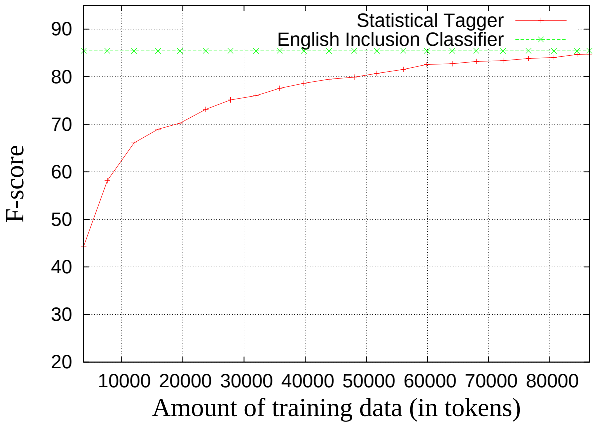


Figure 1: Learning curve of a ML classifier versus the English inclusion classifier's performance.

4 Experiments

The primary focus of this paper is to apply the English inclusion classifier to the German TIGER treebank (Brants et al., 2002) and to evaluate the classifier on a standard NLP task, namely parsing. The aim is to investigate the occurrence of English inclusions in more general newspaper text, and to examine if the detection of English inclusions can improve parsing performance.

The TIGER treebank is a bracketed corpus consisting of 40,020 sentences of newspaper text. The English inclusion classifier was run once over the entire TIGER corpus. In total, the system detected English inclusions in 2,948 of 40,020 sentences (7.4%), 596 of which contained at least one multi-word inclusion. This subset of 596 sentences is the focus of the work reported in the remainder of this paper, and will be referred to as the inclusion set.

A gold standard parse tree for a sentence containing a typical multi-word English inclusion is illustrated in Figure 2. The tree is relatively flat, which is a trait trait of TIGER treebank annotation (Brants et al., 2002). The non-terminal nodes of the tree represent the phrase categories, and the edge labels the grammatical functions. In the example sentence, the English inclusion is contained in a proper noun (*PN*) phrase with a grammatical function of type noun kernel element (*NK*). Each terminal node is POS-tagged as a named entity (*NE*) with the grammatical function ot type proper noun component (*PNC*).

4.1 Data

Two different data sets are used in the experiments: (1) the inclusion set, i.e., the sentences containing multi-word English inclusions recognized by the inclusion classifier, and (2) a stratified sample of sentences randomly extracted from the TIGER corpus, with strata for different sentence lengths. The strata were chosen so that the sentence length distribution of the random set matches that of the inclusion set. The average sentence length of this random set and the inclusion set is therefore the same at 28.4 tokens. This type of sampling is necessary as the inclusion set has a higher average sentence length than a random sample of sentences from TIGER, and because parsing accuracy is correlated with sentence length. Both the inclusion set and the random set consist of 596 sentences and do not overlap.

4.2 Parser

The parsing experiments were performed with a state-of-the-art parser trained on the TIGER corpus which returns both phrase categories and grammatical functions (Dubey, 2005b). Following Klein and Manning (2003), the parser uses an unlexicalized probabilistic context-free grammar (PCFG) and relies on treebank transformations to increase parsing accuracy. Crucially, these transformations make use of TIGER's grammatical functions to relay pertinent lexical information from lexical elements up into the tree.

The parser also makes use of suffix analysis. However, beam search or smoothing are not employed. Based upon an evaluation on the NEGRA treebank (Skut et al., 1998), using a 90%-5%-5% training-development-test split, the parser performs with an accuracy of 73.1 F-score on labelled brackets with a coverage of 99.1% (Dubey, 2005b). These figures were derived on a test set limited to sentences containing 40 tokens or less. In the data set used in this paper, however, sentence length is not limited. Moreover, the average sentence length of our test sets is considerably higher than that of the NEGRA test set. Consequently, a slightly lower performance and/or coverage is anticipated, albeit the type and domain as well as the annotation of both the NEGRA and the TIGER treebanks are very similar. The minor annotation differences that do exist between NEGRA and TIGER are explained in Brants et. al (2002).

4.3 Parser Modifications

We test several variations of the parser. The **baseline** parser does not treat foreign inclusions in any special way: the parser attempts to guess the POS tag and grammatical function labels of the word using the same suffix analysis as for rare or unseen German words. The additional versions of the parser are inspired by the hypothesis that inclusions make parsing difficult, and this difficulty arises primarily because the parser cannot detect inclusions properly. Therefore, a suitable upper bound is to give the parser **perfect tagging** information. Two further versions interface with our inclusion classifier and treat words marked as inclusions differently from native words. The first version does so on a **word-by-word** basis. In contrast, the **inclusion entity** approach attempts to group inclusions, even if a grouping is not posited by phrase structure rules. We now describe each version in more detail.

In the TIGER annotation, preterminals include both POS tags and grammatical function labels. For example, rather than a preterminal node having the category *PRELS* (personal pronoun), it is given the category *PRELS-OA* (accusative personal pronoun). Due to these grammatical function tags, the perfect tagging parser may disambiguate more syntactic information than provided with POS tags alone. Therefore, to make this model more realistic, the parser is required to guess grammatical functions (allowing it to, for example, mistakenly tag an accusative pronoun as nominative, dative or genitive). This gives the parser information about the POS tags of English inclusions (along with other words), but does not give any additional hints about the syntax of the sentence.

The two remaining models both take advantage

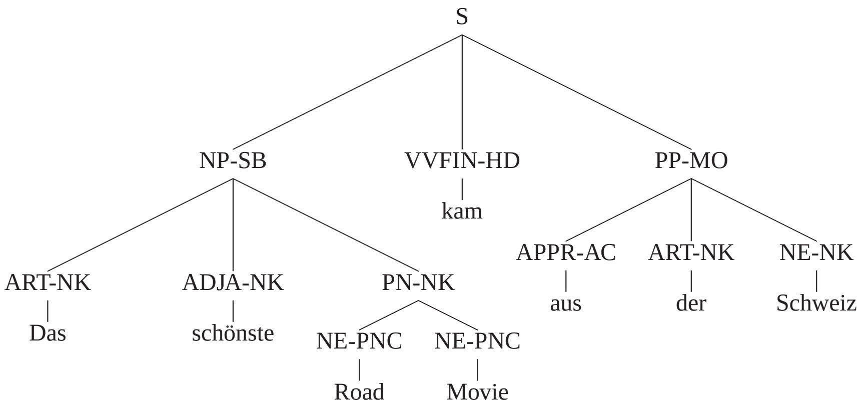
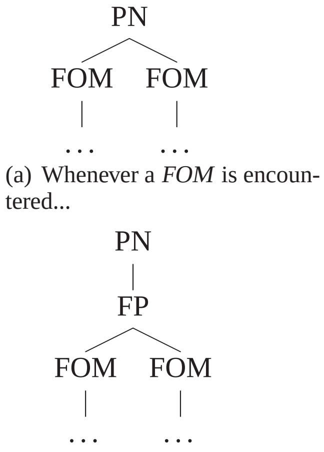


Figure 2: Example parse tree of a German TIGER sentence containing an English inclusion. Translation: The nicest road movie came from Switzerland.

NE	*FM*	*NN*	*KON*	*CARD*	*ADJD*	*APPR*
1185	512	44	8	8	1	1

Table 2: POS tags of foreign inclusions.



(a) Whenever a *FOM* is encountered...

(b) ...a new *FP* category is created

Figure 3: Tree transformation employed in the *inclusion entity* parser.

of information from the inclusion detector. To interface the detector with the parser, we simply mark any inclusion with a special *FOM* (foreign material) tag. The word-by-word parser attempts to guess POS tags itself, much like the baseline. However, whenever it encounters a *FOM* tag, it restricts itself to the set of POS tags observed in inclusions during training (the tags listed in Table 2). When a *FOM* is detected, these and only these POS tags are guessed; all other aspects of the parser remain the same.

The word-by-word parser fails to take advantage of one important trend in the data: that foreign inclusion tokens tend to be adjacent, and these adjacent words usually refer to the same entity. There is nothing stopping the word-by-word parser from positing a constituent boundary between two adjacent foreign inclusions. The inclusion entity model was developed to restrict such spurious bracketing. It does so by way of another tree transformation. The new category *FP* (foreign phrase) is added below any node dominating at least one token marked *FOM* during training. For example, when encountering a *FOM* sequence dominated by *PN* as in Figure 3(a), the tree is modified so that it is the *FP* rule which generates the *FOM* tokens. Figure 3(b) shows the modified tree. In all cases, a unary rule *PN*→FP is introduced. As this extra rule decreases the probability of the entire tree, the parser has a bias to introduce as few of these rules as possible – thus limiting the number of categories which expand to *FOM*s. Once a candidate parse is created during testing, the inverse operation is applied, removing the *FP* node.

4.4 Method

For all experiments reported in this paper, the parser is trained on the TIGER treebank. As the inclusion and random sets are drawn from the whole TIGER treebank, it is necessary to ensure that the data used to train the parser does not overlap with these test sentences. The experiments are therefore designed as multifold cross-validation tests. Using 5 folds, each model is trained on 80% of the data while the remaining 20% are held out. The held out set is then

Data	P	R	F	Dep.	Cov.	AvgCB	0CB	≤2CB
Baseline model								
Inclusion set	56.1	62.6	59.2	74.9	99.2	2.1	34.0	69.0
Random set	63.3	67.3	65.2	81.1	99.2	1.6	40.4	75.1
Perfect tagging model								
Inclusion set	61.3	63.0	62.2	75.1	92.7	1.7	41.5	72.6
Random set	65.8	68.9	67.3	82.4	97.7	1.4	45.9	77.1
Word-by-word model								
Inclusion set	55.6	62.8	59.0	73.1	99.2	2.1	34.2	70.2
Random set	63.3	67.3	65.2	81.1	99.2	1.6	40.4	75.1
Inclusion entity model								
Inclusion set	61.3	65.9	63.5	78.3	99.0	1.7	42.4	77.1
Random set	63.4	67.5	65.4	80.8	99.2	1.6	40.1	75.7

Table 3: Baseline and perfect tagging for inclusion and random sets and results for the word-by-word and the inclusion entity models.

intersected with the inclusion set (or, respectively, the random set). The evaluation metrics are calculated on this subset of the inclusion set (or random set), using the parser trained on the corresponding training data. This process ensures that the test sentences are not contained in the training data.

The overall performance metrics of the parser are calculated on the aggregated totals of the five held out test sets. For each experiment, we report parsing performance in terms of the standard PARSEVAL scores (Abney et al., 1991), including coverage (Cov), labeled precision (P) and recall (R), F-score, the average number of crossing brackets (AvgCB), and the percentage of sentences parsed with zero and with two or fewer crossing brackets (0CB and ≤2CB). In addition, we also report dependency accuracy (Dep), calculated using the approach described in Lin (1995), using the head-picking method used by Dubey (2005a). The labeled bracketing figures (P, R and F), and the dependency score are calculated on all sentences, with those which are out-of-coverage getting zero nodes. The crossing bracket scores are calculated only on those sentences which are successfully parsed.

4.5 Baseline and Perfect Tagging

The baseline, for which the unmodified parser is used, achieves a high coverage at over 99% for both the inclusion and the random sets (see Table 3). However, scores differ for the bracketing measures. Using stratified shuffling[1], we performed a *t*-test on precision and recall, and found both to be significantly worse in the inclusion condition. Overall, the harmonic mean (F) of precision and recall was 65.2 on the random set, 6 points better than 59.2 F observed on the inclusion set. Similarly, dependency and cross-bracketing scores are higher on the random test set. This result strongly indicates that sentences containing English inclusions present difficulty for the parser, compared to length-matched sentences without inclusions.

When providing the parser with perfect tagging information, scores improve both for the inclusion and the random TIGER samples, resulting in F-scores of 62.2 and 67.3, respectively. However, the coverage for the inclusion set decreases to 92.7% whereas the coverage for the random set is 97.7%. In both cases, the lower coverage is caused by the parser being forced to use infrequent tag sequences, with the much lower coverage of the inclusion set likely due to infrequent tags (notable *FM*), solely associated with inclusions. While perfect tagging increases overall accuracy, a difference of 5.1 in F-score remains between the random and inclusion test sets. Although smaller than that of the baseline runs, this difference shows that even with perfect tagging,

[1] This approach to statistical testing is described in: `http://www.cis.upenn.edu/~dbikel/software.html`

parsing English inclusions is harder than parsing monolingual data.

So far, we have shown that the English inclusion classifier is able to detect sentences that are difficult to parse. We have also shown that perfect tagging helps to improve parsing performance but is insufficient when it comes to parsing sentences containing English inclusions. In the next section, we will examine how the knowledge provided by the English inclusion classifier can be exploited to improve parsing performance for such sentences.

4.6 Word-by-word Model

The word-by-word model achieves the same coverage on the inclusion set as the baseline but with a slightly lower F of 59.0. All other scores, including dependency accuracy and cross bracketing results are similar to those of the baseline (see Table 3). This shows that limiting the parser's choice of POS tags to those encountered for English inclusions is not sufficient to deal with such constructions correctly. In the error analysis presented in Section 5, we report that the difficulty in parsing multi-word English inclusions is recognizing them as constituents, rather than recognizing their POS tags. We attempt to overcome this problem with the inclusion entity model.

4.7 Inclusion Entity Model

The inclusion entity parser attains a coverage of 99.0% on the inclusion set, similiar to the coverage of 99.2% obtained by the baseline model on the same data. On all other measures, the inclusion entity model exceeds the performance of the baseline, with a precision of 61.3% (5.2% higher than the baseline), a recall of 65.9% (3.3% higher), an F of 63.5 (4.3 higher) and a dependency accuracy of 78.3% (3.4% higher). The average number of crossing brackets is 1.7 (0.4 lower), with 42.4% of the parsed sentences having no crossing brackets (8.2% higher), and 77.1% having two or fewer crossing brackets (8.1% higher). When testing the inclusion entity model on the random set, the performance is very similar to the baseline model on this data. While coverage is the same, F and cross-brackting scores are marginally improved, and the dependency score is marginally deteriorated. This shows that the inclusion entity model does not harm the parsing accuracy of sentences that do not actually contain foreign inclusions.

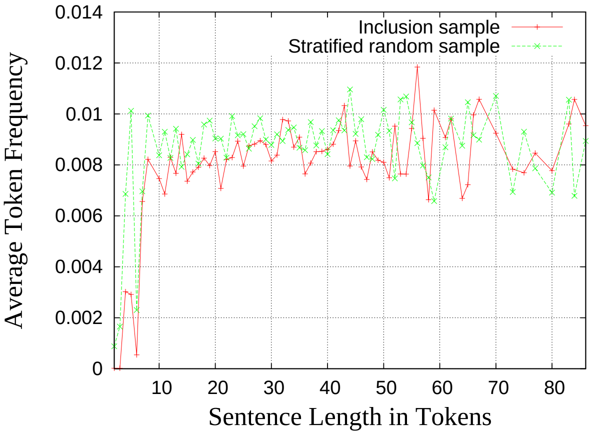


Figure 4: Average relative token frequencies for sentences of equal length.

Not only did the inclusion entity parser perform above the baseline on every metric for the inclusion set, its performance also exceeds that of the perfect tagging model on all measures except precision and average crossing brackets, where both models are tied. These results clearly indicate that the inclusion entity model is able to leverage the additional information about English inclusions provided by our inclusion classifier. However, it is also important to note that the performance of this model on the inclusion set is still consistently lower than that of all models on the random set. This demonstrates that sentences with inclusions are more difficult to parse than monolingual sentences, even in the presence of information about the inclusions that the parser can exploit.

Comparing the inclusion set to the length-matched random set is arguably not entirely fair as the latter may not contain as many infrequent tokens as the inclusion set. Figure 4 shows the average relative token frequencies for sentences of equal length for both sets. The frequency profiles of the two data sets are broadly similar (the difference in means of both groups is only 0.000676), albeit significantly different according to a paired t-test ($p \leq 0.05$). This is one reason why the inclusion entity model's performance on the inclusion set does not reach the upper limit set by the random sample.

Phrase cat.	Frequency	Example
PN	91	The Independent
CH	10	Made in Germany
NP	4	Peace Enforcement
CNP	2	Botts and Company
–	2	Chief Executives

Table 4: Gold phrase categories of inclusions.

Phrase bracket (PB) frequency	BL	IE
$PB_{PRED} > PB_{GOLD}$	62%	51%
$PB_{PRED} < PB_{GOLD}$	11%	13%
$PB_{PRED} = PB_{GOLD}$	27%	36%

Table 5: Bracket frequency of the predicted baseline (BL) and inclusion entity (IE) model output compared to the gold standard.

5 Error Analysis

The error analysis is limited to 100 sentences selected from the inclusion set parsed with both the baseline and the inclusion entity model. This sample contains 109 English inclusions, five of which are false positives, i.e., the output of the English inclusion classifier is incorrect. The precision of the classifier in recognizing multi-word English inclusions is therefore 95.4% for this TIGER sample.

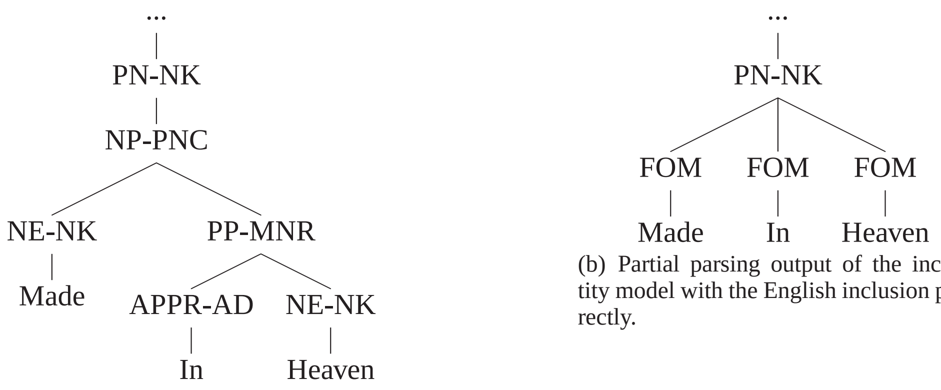
Table 4 illustrates that the majority of multi-word English inclusions are contained in a proper noun (*PN*) phrase, including names of companies, political parties, organizations, films, newspapers, etc. A less frequent phrasal category is chunk (*CH*) which tends to be used for slogans, quotes or expressions like *Made in Germany*. Even in this small sample, annotations of inclusions as either *PN* or *CH*, and not the other, can be misleading. For example, the organization *Friends of the Earth* is annotated as a *PN*, whereas another organization *International Union for the Conservation of Nature* is marked as a *CH* in the gold standard. This suggests that the annotation guidelines on foreign inclusions could be improved when differentiating between phrase categories containing foreign material.

For the majority of sentences (62%), the baseline model predicts more brackets than are present in the gold standard parse tree (see Table 5). This number decreases by 11% to 51% when parsing with the inclusion entity model. This suggests that the baseline parser does not recognize English inclusions as constituents, and instead parses their individual tokens as separate phrases. Provided with additional information of multi-word English inclusions in the training data, the parser is able to overcome this problem.

We now turn our attention to how accurately the various parsers are at predicting both phrase bracketing and phrase categories (see Table 6). For 46 (42.2%) of inclusions, the baseline model makes an error with a negative effect on performance. In 39 cases (35.8%), the phrase bracketing and phrase category are incorrect, and constituent boundaries occur within the inclusion, as illustrated in Figure 5(a). Such errors also have a detrimental effect on the parsing of the remainder of the sentence. Overall, the baseline model predicts the correct phrase bracketing and phrase category for 63 inclusions (57.8%). Conversely, the inclusion entity model, which is given information on tag consistency within inclusions via the *FOM* tags, is able to determine the correct phrase bracketing and phrase category for 67.9% inclusions (10.1% more), e.g. see Figure 5(b). Both the phrase bracketing and phrase category are predicted incorrectly in only 6 cases (5.5%). The inclusion entity model's improved phrase boundary prediction for 31 inclusions (28.4% more correct) is likely to have an overall positive effect on the parsing decisions made for the context which they appear in. Nevertheless, the inclusion entity parser still has difficulty determining the correct phrase category in 25 cases (22.9%). The main confusion lies between assigning the categories *PN*, *CH* and *NP*, the most frequent phrase categories of multi-word English inclusions. This is also partially due to the ambiguity between these phrases in the gold standard. Finally, few parsing errors (4) are caused by the inclusion entity parser due to the markup of false positive inclusions (mainly boundary errors).

6 Discussion and Conclusion

This paper has argued that English inclusions in German text is an increasingly pervasive instance of language mixing. Starting with the hypothesis that such inclusions can be a significant source of errors for monolingual parsers, we found evidence that an unmodified state-of-the-art parser for Ger-



(a) Partial parsing output of the baseline model with a constiuent boundary in the English inclusion.

(b) Partial parsing output of the inclusion entity model with the English inclusion parsed correctly.

Figure 5: Comparing baseline model output to inclusion entity model output.

Errors	No. of inclusions (in %)
Parser: baseline model, data: inclusion set	
Incorrect PB and PC	39 (35.8%)
Incorrect PC	5 (4.6%)
Incorrect PB	2 (1.8%)
Correct PB and PC	63 (57.8%)
Parser: inclusion entity model, data: inclusion set	
Incorrect PB and PC	6 (5.5%)
Incorrect PC	25 (22.9%)
Incorrect PB	4 (3.7%)
Correct PB and PC	74 (67.9%)

Table 6: Baseline and inclusion entity model errors for inclusions with respect to their phrase bracketing (PB) and phrase category (PC).

man performs substantially worse on a set of sentences with English inclusions compared to a set of length-matched sentences randomly sampled from the same corpus. The lower performance on the inclusion set persisted even when the parser when given gold standard POS tags in the input.

To overcome the poor accuracy of parsing inclusions, we developed two methods for interfacing the parser with an existing annotation-free inclusion detection system. The first method restricts the POS tags for inclusions that the parser can assign to those found in the data. The second method applies tree transformations to ensure that inclusions are treated as phrases. An evaluation on the TIGER corpus shows that the second method yields a performance gain of 4.3 in F-score over a baseline of no inclusion detection, and even outperforms a model involving perfect POS tagging of inclusions.

To summarize, we have shown that foreign inclusions present a problem for a monolingual parser. We also demonstrated that it is insufficient to know where inclusions are or even what their parts of speech are. Parsing performance only improves if the parser also has knowledge about the structure of the inclusions. It is particularly important to know when adjacent foreign words are likely to be part of the same phrase. As our error analysis showed, this prevents cascading errors further up in the parse tree.

Finally, our results indicate that future work could improve parsing performance for inclusions further: we found that parsing the inclusion set is still harder than parsing a randomly sampled test set, even for our best-performing model. This provides an upper bound on the performance we can expect from a parser that uses inclusion detection. Future work will also involve determining the English inclusion classifier's merit when applied to rule-based parsing.

Acknowledgements

This research is supported by grants from the Scottish Enterprise Edinburgh-Stanford Link (R36759), ESRC, and the University of Edinburgh. We would also like to thank Claire Grover for her comments and feedback.

References

Steven Abney, Dan Flickenger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In Ezra Black, editor, *HLT'91: Proceedings of the workshop on Speech and Natural Language*, pages 306–311, Morristown, NJ, USA. Association for Computational Linguistics.

Beatrice Alex. 2005. An unsupervised system for identifying English inclusions in German text. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), Student Research Workshop*, pages 133–138, Ann Arbor, Michigan, USA.

Beatrice Alex. 2006. Integrating language knowledge resources to extend the English inclusion classifier to a new language. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Gisle Andersen. 2005. Assessing algorithms for automatic extraction of Anglicisms in Norwegian texts. In *Corpus Linguistics 2005*, Birmingham, UK.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories (TLT02)*, pages 24–41, Sozopol, Bulgaria.

Amit Dubey. 2005a. *Statistical Parsing for German: Modeling syntactic properties and annotation differences*. Ph.D. thesis, Saarland University, Germany.

Amit Dubey. 2005b. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 314–321, Ann Arbor, Michigan, USA.

Jenny Finkel, Shipra Dingare, Christopher D. Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: Gene and protein identification in biomedical text. *BMC Bioinformatics*, 6(Suppl 1):S5.

Martin Forst and Ronald M. Kaplan. 2006. The importance of precise tokenizing for deep grammars. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 369–372, Genoa, Italy.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 423–430, Saporo, Japan.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-03)*, pages 180–183, Edmonton, Canada.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1420–1425, Montreal, Canada.

Jean-Christophe Marcadet, Volker Fischer, and Claire Waast-Richard. 2005. A transformation-based learning approach to language identification for mixed-lingual text-to-speech synthesis. In *Proceedings of Interspeech 2005 - ICSLP*, pages 2249–2252, Lisbon, Portugal.

Beat Pfister and Harald Romsdorfer. 2003. Mixed-lingual analysis for polyglot TTS synthesis. In *Proceedings of Eurospeech 2003*, pages 2037–2040, Geneva, Switzerland.

Ariel Schwartz and Marti Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on Biocomputing (PSB 2003)*, pages 451–462, Kauai, Hawaii.

Wojciech Skut, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of German newspaper text. In *Proceedings of the Conference on Language Resources and Evaluation (LREC 1998)*, pages 705–712, Granada, Spain.

Boye Wangensteen. 2002. Nettbasert nyordsinnsamling. *Språknytt*, 2:17–19.

LEDIR: An Unsupervised Algorithm for Learning Directionality of Inference Rules

Rahul Bhagat, Patrick Pantel, Eduard Hovy
Information Sciences Institute
University of Southern California
Marina del Rey, CA
{rahul,pantel,hovy}@isi.edu

Abstract

Semantic inference is a core component of many natural language applications. In response, several researchers have developed algorithms for automatically learning inference rules from textual corpora. However, these rules are often either imprecise or underspecified in directionality. In this paper we propose an algorithm called LEDIR that filters incorrect inference rules and identifies the directionality of correct ones. Based on an extension to Harris's distributional hypothesis, we use selectional preferences to gather evidence of inference directionality and plausibility. Experiments show empirical evidence that our approach can classify inference rules significantly better than several baselines.

1 Introduction

Paraphrases are textual expressions that convey the same meaning using different surface forms. Textual entailment is a similar phenomenon, in which the presence of one expression licenses the validity of another. Paraphrases and inference rules are known to improve performance in various NLP applications like Question Answering (Harabagiu and Hickl 2006), summarization (Barzilay et al. 1999) and Information Retrieval (Anick and Tipirneni 1999).

Paraphrase and entailment involve inference rules that license a conclusion when a premise is given. Deciding whether a proposed inference rule is fully valid is difficult, however, and most NL systems instead focus on plausible inference. In this case, one statement has some likelihood of being identical in meaning to, or derivable from, the other. In the rest of this paper we discuss plausible inference only.

Given the importance of inference, several researchers have developed inference rule collections. While manually built resources like WordNet (Fellbaum 1998) and Cyc (Lenat 1995) have been around for years, for coverage and domain adaptability reasons many recent approaches have focused on automatic acquisition of paraphrases (Barzilay and McKeown 2001) and inference rules (Lin and Pantel 2001; Szpektor et al. 2004). The downside of these approaches is that they often result in incorrect inference rules or in inference rules that are underspecified in directionality (i.e. asymmetric but are wrongly considered symmetric). For example, consider an inference rule from DIRT (Lin and Pantel 2001):

$$X\ eats\ Y \Leftrightarrow X\ likes\ Y \qquad (1)$$

All rules in DIRT are considered symmetric. Though here, one is most likely to infer that "*X eats Y*" ⇒ "*X likes Y*", because if someone eats something, he most probably likes it[1], but if he likes something he might not necessarily be able to eat it. So for example, given the sentence "*I eat spicy food*", one is mostly likely to infer that "*I like spicy food*". On the other hand, given the sentence "*I like rollerblading*", one cannot infer that "*I eat rollerblading*".

In this paper, we propose an algorithm called **LEDIR** (pronounced "leader") for **LE**arning **Di**rectionality of **I**nference **R**ules. Our algorithm filters incorrect inference rules and identifies the directionality of the correct ones. Our algorithm

[1] There could be certain usages of "*X eats Y*" where, one might not be able to infer "*X likes Y*" (for example metaphorical). But, in most cases, this inference holds.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 161–170, Prague, June 2007. ©2007 Association for Computational Linguistics

works with any resource that produces inference rules of the form shown in example (1). We use both the distributional hypothesis and selectional preferences as the basis for our algorithm. We provide empirical evidence to validate the following main contribution:

***Claim**: Relational selectional preferences can be used to automatically determine the plausibility and directionality of an inference rule.*

2 Related Work

In this section, we describe applications that can benefit by using inference rules and their directionality. We then talk about some previous work in this area.

2.1 Applications

Open domain question answering approaches often cast QA as the problem of finding some kind of semantic inference between a question and its answer(s) (Moldovan et al. 2003; Echiabi and Marcu 2003). Harabagiu and Hickl (2006) recently demonstrated that textual entailment inference information, which in this system is a set of directional inference relations, improves the performance of a QA system significantly even without using any other form of semantic inference. This evidence supports the idea that learning the directionality of other sets of inference rules may improve QA performance.

In Multi-Document Summarization (MDS), paraphrasing is useful for determining sentences that have similar meanings (Barzilay et al. 1999). Knowing the directionality between the inference rules here could allow the MDS system to choose either the more specific or general sentence depending on the purpose of the summary.

In IR, paraphrases have been used for query expansion, which is known to promote effective retrieval (Anick and Tipirneni 1999). Knowing the directionality of rules here could help in making a query more general or specific depending on the user needs.

2.2 Learning Inference Rules

Automatically learning paraphrases and inference rules from text is a topic that has received much attention lately. Barzilay and McKeown (2001) for paraphrases, DIRT (Lin and Pantel 2001) and TEASE (Szpektor et al. 2004) for inference rules, are recent approaches that have achieved promising results. While all these approaches produce collections of inference rules that have good recall, they suffer from the complementary problem of low precision. They also make no attempt to distinguish between symmetric and asymmetric inference rules. Given the potential positive impact shown in Section 2.1 of learning the directionality of inference rules, there is a need for methods, such as the one we present, to improve existing automatically created resources.

2.3 Learning Directionality

There have been a few approaches at learning the directionality of restricted sets of semantic relations, mostly between verbs. Chklovski and Pantel (2004) used lexico-syntactic patterns over the Web to detect certain types of symmetric and asymmetric relations between verbs. They manually examined and obtained lexico-syntactic patterns that help identify the types of relations they considered and used these lexico-syntactic patterns over the Web to detect these relations among a set of candidate verb pairs. Their approach however is limited only to verbs and to specific types of verb-verb relations.

Zanzotto et al. (2006) explored a selectional preference-based approach to learn asymmetric inference rules between verbs. They used the selectional preferences of a single verb, i.e. the semantic types of a verb's arguments, to infer an asymmetric inference between the verb and the verb form of its argument type. Their approach however applies also only to verbs and is limited to some specific types of verb-argument pairs.

Torisawa (2006) presented a method to acquire inference rules with temporal constraints, between verbs. They used co-occurrences between verbs in Japanese coordinated sentences and co-occurrences between verbs and nouns to learn the verb-verb inference rules. Like the previous two methods, their approach too deals only with verbs and is limited to learning inference rules that are temporal in nature.

Geffet and Dagan (2005) proposed an extension to the distributional hypothesis to discover entailment relation between words. They model the context of a word using its syntactic features and compare the contexts of two words for strict inclusion to infer lexical entailment. In principle, their work is the most similar to ours. Their method however is limited to lexical entailment and they show its effectiveness for nouns. Our method on the other hand deals with inference rules between binary relations and includes inference rules between verbal relations, non-verbal relations and multi-word relations. Our definition of context and the methodology for obtaining context similarity and overlap is also much different from theirs.

3 Learning Directionality of Inference Rules

The aim of this paper is to filter out incorrect inference rules and to identify the directionality of the correct ones.

Let $p_i \Leftrightarrow p_j$ be an inference rule where each p is a binary semantic relation between two entities x and y. Let $<x, p, y>$ be an instance of relation p.

Formal problem definition: *Given the inference rule $p_i \Leftrightarrow p_j$, we want to conclude which one of the following is more appropriate:*

1. $p_i \Leftrightarrow p_j$
2. $p_i \Rightarrow p_j$
3. $p_i \Leftarrow p_j$
4. *No plausible inference*

Consider the example (1) from section 1. There, it is most plausible to conclude "*X eats Y*" ⇒ "*X likes Y*".

Our algorithm LEDIR uses selectional preferences along the lines of Resnik (1996) and Pantel et al. (2007) to determine the plausibility and directionality of inference rules.

3.1 Underlying Assumption

Many approaches to modeling lexical semantics have relied on the distributional hypothesis (Harris 1954), which states that words that appear in the same contexts tend to have similar meanings. The idea is that context is a good indicator of a word meaning. Lin and Pantel (2001) proposed an extension to the distributional hypothesis and applied it to paths in dependency trees, where if two paths tend to occur in similar contexts it is hypothesized that the meanings of the paths tend to be similar.

In this paper, we assume and propose a further extension to the distributional hypothesis and call it the "Directionality Hypothesis".

Directionality Hypothesis: *If two binary semantic relations tend to occur in similar contexts and the first one occurs in significantly more contexts than the second, then the second most likely implies the first and not vice versa.*

The intuition here is that of generality. The more general a relation, more the types (and number) of contexts in which it is likely to appear. Consider the example (1) from section 1. The fact is that there are many more things that someone might like than those that someone might eat. Hence, by applying the directionality hypothesis, one can infer that "*X eats Y*" ⇒ "*X likes Y*".

The key to applying the distributional hypothesis to the problem at hand is to model the contexts appropriately and to introduce a measure for calculating context similarity. Concepts in semantic space, due to their abstractive power, are much richer for reasoning about inferences than simple surface words. Hence, we model the context of a relation p of the form $<x, p, y>$ by using the semantic classes *C(x)* and *C(y)* of words that can be instantiated for x and y respectively. To measure context similarity of two relations, we calculate the overlap coefficient (Manning and Schütze, 1999) between their contexts.

3.2 Selectional Preferences

The selectional preferences of a predicate is the set of semantic classes that its arguments can belong to (Wilks 1975). Resnik (1996) gave an information theoretical formulation of the idea. Pantel et al. (2007) extended this idea to non-verbal relations by defining the relational selectional preferences (RSPs) of a binary relation p as the set of semantic classes $C(x)$ and $C(y)$ of words that can occur in positions x and y respectively.

The set of semantic classes $C(x)$ and $C(y)$ can be obtained either from a manually created taxonomy like WordNet as proposed in the above previous approaches or by using automatically generated classes from the output of a word clustering algorithm as proposed in Pantel et al. (2007). For example given a relation like "*X likes Y*", its RSPs from WordNet could be *{individual, social_group…}* for *X* and *{individual, food, activity…}* for *Y*.

In this paper, we deployed both the Joint Relational Model (JRM) and Independent Relational Model (IRM) proposed by Pantel et al. (2007) to obtain the selectional preferences for a relation p.

Model 1: Joint Relational Model (JRM)

The JRM uses a large corpus to learn the selectional preferences of a binary semantic relation by considering its arguments jointly.

Given a relation *p* and large corpus of English text, we first find all occurrences of relation *p* in the corpus. For every instance <*x, p, y*> in the corpus, we obtain the sets *C*(*x*) and *C*(*y*) of the semantic classes that *x* and *y* belong to. We then accumulate the frequencies of the triples <*c*(*x*), *p*, *c*(*y*)> by assuming that every $c(x) \in C(x)$ can co-occur with every $c(y) \in C(y)$ and vice versa. Every triple <*c*(*x*), *p*, *c*(*y*)> obtained in this manner is a candidate selectional preference for *p*. Following Pantel et al. (2007), we rank these candidates using Pointwise mutual information (Cover and Thomas 1991). The ranking function is defined as the strength of association between two semantic classes, c_x and c_y[2], given the relation *p*:

$$pmi\left(c_x|p; c_y|p\right) = \log \frac{P\left(c_x, c_y|p\right)}{P\left(c_x|p\right)P\left(c_y|p\right)} \tag{3.1}$$

Let $|c_x, p, c_y|$ denote the frequency of observing the instance <*c*(*x*), *p*, *c*(*y*)>. We estimate the probabilities of Equation 3.1 using maximum likelihood estimates over our corpus:

$$P\left(c_x|p\right) = \frac{|c_x, p, *|}{|*, p, *|} \quad P\left(c_y|p\right) = \frac{|*, p, c_y|}{|*, p, *|} \tag{3.2}$$
$$P\left(c_x, c_y|p\right) = \frac{|c_x, p, c_y|}{|*, p, *|}$$

We estimate the above frequencies using:

$$|c_x, p, *| = \sum_{w \in c_x} \frac{|w, p, *|}{|C(w)|} \quad |*, p, c_y| = \sum_{w \in c_y} \frac{|*, p, w|}{|C(w)|} \tag{3.3}$$
$$|c_x, p, c_y| = \sum_{w_1 \in c_x, w_2 \in c_y} \frac{|w_1, p, w_2|}{|C(w_1)| \times |C(w_2)|}$$

where |*x*, *p*, *y*| denotes the frequency of observing the instance <*x*, *p*, *y*> and |C(*w*)| denotes the number of classes to which word *w* belongs. |C(*w*)| distributes *w*'s mass equally among all of its senses C(*w*).

Model 2: Independent Relational Model (IRM)

Due to sparse data, the JRM is likely to miss some pair(s) of valid relational selectional preferences. Hence we use the IRM, which models the arguments of a binary semantic relation independently.

Similar to JRM, we find all instances of the form <*x, p, y*> for a relation *p*. We then find the sets *C*(*x*) and *C*(*y*) of the semantic classes that *x* and *y* belong to and accumulate the frequencies of the triples <*c*(*x*), *p*, *> and <*, *p*, *c*(*y*)> where $c(x) \in C(x)$ and $c(y) \in C(y)$.

All the tuples <*c*(*x*), *p*, *> and <*, *p*, *c*(*y*)> are the independent candidate RSPs for a relation *p* and we rank them according to equation 3.3.

Once we have the independently learnt RSPs, we need to convert them into a joint representation for use by the inference plausibility and directionality model. To do this, we obtain the Cartesian product between the sets <*C*(*x*), *p*, *> and <*, *p*, *C*(*y*)> for a relation *p*. The Cartesian product between two sets A and B is given by:

$$A \times B = \left\{(a,b) : \forall a \in A \quad and \quad \forall b \in B\right\} \tag{3.4}$$

Similarly we obtain:

$$\langle C_x, p, * \rangle \times \langle *, p, C_y \rangle = \left\{ \begin{matrix} \langle c_x, p, c_y \rangle : \ \forall \langle c_x, p, * \rangle \in \langle C_x, p, * \rangle \quad and \\ \forall \langle *, p, c_y \rangle \in \langle *, p, C_y \rangle \end{matrix} \right\} \tag{3.5}$$

The Cartesian product in equation 3.5 gives the joint representation of the RSPs of the relation *p* learnt using IRM. In the joint representation, the IRM RSPs have the form <*c*(*x*), *p*, *c*(*y*)> which is the same form as the JRM RSPs.

3.3 Inference plausibility and directionality model

Our model for determining inference plausibility and directionality is based on the intuition that for an inference to hold between two semantic relations there must exist sufficient overlap between their contexts and the directionality of the inference depends on the quantitative comparison between their contexts.

Here we model the context of a relation by the selectional preferences of that relation. We determine the plausibility of an inference based on the overlap coefficient (Manning and Schütze, 1999) between the selectional preferences of the two paths. We determine the directionality based on the difference in the number of selectional preferences of the relations when the inference seems plausible.

Given a candidate inference rule $p_i \Leftrightarrow p_j$, we first obtain the RSPs <*C*(*x*), p_i, *C*(*y*)> for p_i and <*C*(*x*), p_j, *C*(*y*)> for p_j. We then calculate the overlap coefficient between their respective RSPs. Overlap coefficient is one of the many distribu-

[2] c_x and c_y are shorthand for *c*(*x*) and *c*(*y*) in our equations.

tional similarity measures used to calculate the similarity between two vectors A and B:

$$sim(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)} \qquad (3.6)$$

The overlap coefficient between the selectional preferences of p_i and p_j is calculated as:

$$sim(p_i,p_j) = \frac{|\langle C_x,p_i,C_y\rangle \cap \langle C_x,p_j,C_y\rangle|}{\min(|C_x,p_i,C_y|,|C_x,p_j,C_y|)} \qquad (3.7)$$

If sim(p_i,p_j) is above a certain empirically determined threshold α (≤1), we conclude that the inference is plausible, i.e.:

```
If  $sim(p_i,p_j) \geq \alpha$
        we conclude the inference is plausible
else
        we conclude the inference is not plausible
```

For a plausible inference, we then compute the ratio between the number of selectional preferences $|C(x), p_i, C(y)|$ for p_i and $|C(x), p_j, C(y)|$ for p_j and compare it against an empirically determined threshold β (≥1) to determine the direction of inference. So the algorithm is:

```
If       $\frac{|C_x,p_i,C_y|}{|C_x,p_j,C_y|} \geq \beta$     we conclude $p_i \Leftarrow p_j$
else if  $\frac{|C_x,p_i,C_y|}{|C_x,p_j,C_y|} \leq \frac{1}{\beta}$     we conclude $p_i \Rightarrow p_j$
else                                 we conclude $p_i \Leftrightarrow p_j$
```

4 Experimental Setup

In this section, we describe our experimental setup to validate our claim that LEDIR can be used to determine plausibility and directionality of an inference rule.

Given an inference rule of the form $p_i \Leftrightarrow p_j$, we want to use automatically learned relational selectional preferences to determine whether the inference rule is valid and if it is valid then what its directionality is.

4.1 Inference Rules

LEDIR can work with any set of binary semantic inference rules. For the purpose of this paper, we chose the inference rules from the DIRT resource (Lin and Pantel 2001). DIRT consists of 12 million rules extracted from 1GB of newspaper text (AP Newswire, San Jose Mercury and Wall Street Journal). For example, "*X eats Y*" ⇔ "*X likes Y*" is an inference rule from DIRT.

4.2 Semantic Classes

Appropriate choice of semantic classes is crucial for learning relational selectional preferences. The ideal set should have semantic classes that have the right balance between abstraction and discrimination, the two important characteristics that are often conflicting. A very general class has limited discriminative power, while a very specific class has limited abstractive power. Finding the right balance here is a separate research problem of its own.

Since the ideal set of universally acceptable semantic classes in unavailable, we decided to use the Pantel et al. (2007) approach of using two sets of semantic classes. This approach gave us the advantage of being able to experiment with sets of classes that vary a lot in the way they are generated but try to maintain the granularity by obtaining approximately the same number of classes.

The first set of semantic classes was obtained by running the CBC clustering algorithm (Pantel and Lin, 2002) on TREC-9 and TREC-2002 newswire collections consisting of over 600 million words. This resulted in 1628 clusters, each representing a semantic class.

The second set of semantic classes was obtained by using WordNet 2.1 (Fellbaum 1998). We obtained a cut in the WordNet noun hierarchy[3] by manual inspection and used all the synsets below a cut point as the semantic class at that node. Our inspection showed that the synsets at depth four formed the most natural semantic classes[4]. A cut at depth four resulted in a set of 1287 semantic classes, a set that is much coarser grained than WordNet which has an average depth of 12. This seems to be a depth that gives a reasonable abstraction while maintaining good discriminative power. It would however be interesting to experiment with more sophisticated algorithms for extracting semantic classes from WordNet and see their effect

[3] Since we are dealing with only noun binary relations, we use only WordNet noun Hierarchy.

[4] By natural, here, we simply mean that a manual inspection by the authors showed that, at depth four, the resulting clusters had struck a better granularity balance than other cutoff points. We acknowledge that this is a very coarse way of extracting concepts from WordNet.

on the relational selectional preferences, something we do not address this in this paper.

4.3 Implementation

We implemented LEDIR with both the JRM and IRM models using inference rules from DIRT and semantic classes from both CBC and WordNet. We parsed the 1999 AP newswire collection consisting of 31 million words with Minipar (Lin 1993) and used this to obtain the probability statistics for the models (as described in section 3.2).

We performed both system-wide evaluations and intrinsic evaluations with different values of α and β parameters. Section 5 presents these results and our error analysis.

4.4 Gold Standard Construction

In order to evaluate the performance of the different systems, we compare their outputs against a manually annotated gold standard. To create this gold standard, we randomly sampled 160 inference rules of the form $p_i \Leftrightarrow p_j$ from DIRT. We discarded three rules since they contained nominalizations[5].

For every inference rule of the form $p_i \Leftrightarrow p_j$, the annotation guideline asked annotators (in this paper we used two annotators) to choose the most appropriate of the four options:

1. $p_i \Leftrightarrow p_j$
2. $p_i \Rightarrow p_j$
3. $p_i \Leftarrow p_j$
4. *No plausible inference*

To help the annotators with their decisions, the annotators were provided with 10 randomly chosen instances for each inference rule. These instances, extracted from DIRT, provided the annotators with context where the inference could hold. So for example, for the inference rule "*X eats Y*" ⇔ "*X likes Y*", an example instance would be "*I eat spicy food*" ⇔ "*I like spicy food*". The annotation guideline however gave the annotators the freedom to think of examples other than the ones provided to make their decisions.

The annotators found that while some decisions were quite easy to make, the more complex ones often involved the choice between bi-directionality and one of the directions. To minimize disagreements and to get a better understanding of the task, the annotators trained themselves by annotating several samples together.

We divided the set of 157 inference rules, into a development set of 57 inference rules and a blind test set of 100 inference rules. Our two annotators annotated the development test set together to train themselves. The blind test set was then annotated individually to test whether the task is well defined. We used the kappa statistic (Siegel and Castellan Jr. 1988) to calculate the inter-annotator agreement, resulting in κ=0.63. The annotators then looked at the disagreements together to build the final gold standard.

All this resulted in a final gold standard of 100 annotated DIRT rules.

4.5 Baselines

To get an objective assessment of the quality of the results obtained by using our models, we compared the output of our systems against three baselines:

B-random: *Randomly assigns one of the four possible tags to each candidate inference rule.*

B-frequent: *Assigns the most frequently occurring tag in the gold standard to each candidate inference rule.*

B-DIRT: *Assumes each inference rule is bidirectional and assigns the bidirectional tag to each candidate inference rule.*

5 Experimental Results

In this section, we provide empirical evidence to validate our claim that the plausibility and directionality of an inference rule can be determined using LEDIR.

5.1 Evaluation Criterion

We want to measure the effectiveness of LEDIR for the task of determining the validity and directionality of a set of inference rules. We follow the standard approach of reporting system accuracy by comparing system outputs on a test set with a manually created gold standard. Using the gold standard described in Section 4.4, we measure the accuracy of our systems using the following formula:

[5] For the purpose of simplicity, we in our experiments did not use DIRT rules containing nominalizations. The algorithm however can be applied without change to inference rules containing nominalization. In fact, in the resource that we plan to release soon, we have applied the algorithm without change to DIRT rules containing nominalizations.

$$Accuracy = \frac{|correctly \quad tagged \quad \inf erences|}{|input \quad \inf erences|}$$

5.2 Result Summary

We ran all our algorithms with different parameter combinations on the development set (the 57 DIRT rules described in Section 4.4). This resulted in a total of 420 experiments on the development set. Based on these experiments, we used the accuracy statistic to obtain the best parameter combination for each of our four systems. We then used these parameter values to obtain the corresponding percentage accuracies on the test set for each of the four systems.

Model		α	β	Accuracy (%)
B-random		-	-	25
B-frequent		-	-	34
B-DIRT		-	-	25
JRM	CBC	0.15	2	38
	WN	0.55	2	38
IRM	**CBC**	**0.15**	**3**	**48**
	WN	0.45	2	43

Table 1: Summary of results on the test set

Table 1 summarizes the results obtained on the test set for the three baselines and for each of the four systems using the best parameter combinations obtained as described above. The overall best performing system uses the IRM algorithm with RSPs form CBC. Its performance is found to be significantly better than all the three baselines using the Student's paired t-test (Manning and Schütze, 1999) at $p<0.05$. However, this system is not statistically significant when compared with the other LEDIR implementations (JRM and IRM with WordNet).

5.3 Performance and Error Analysis

The best performing system selected using the development set is the IRM system using CBC with the parameters α=0.15 and β=3. In general, the results obtained on the test set show that the IRM tends to perform better than the JRM. This observation points at the sparseness of data available for learning RSPs for the more restrictive JRM, the reason why we introduced the IRM in the first place. A much larger corpus would be needed to obtain good enough coverage for the JRM.

		GOLD STANDARD			
		⇔	⇒	⇐	NO
SYSTEM	⇔	16	1	3	7
	⇒	0	3	1	3
	⇐	7	4	22	15
	NO	2	3	4	9

Table 2: Confusion Matrix for the best performing system, IRM using CBC with α=0.15 and β=3.

Table 2 shows the confusion matrix for the overall best performing system as selected using the development set (results are taken from the test set). The confusion matrix indicates that the system does a very good job of identifying the directionality of the correct inference rules, but gets a big performance hit from its inability to identify the incorrect inference rules accurately. We will analyze this observation in more detail below.

Figure 1 plots the variation in accuracy of IRM with different RSPs and different values of α and β. The figure shows a very interesting trend. It is clear that for all values of β, systems for IRM using CBC tend to reach their peak in the range 0.15 ≤ α ≤ 0.25, whereas the systems for IRM using WordNet (WN), tend to reach their peak in the range 0.4 ≤ α ≤ 0.6. This variation indicates the kind of impact the selection of semantic classes could have on the overall performance of the system. This is not hard evidence, but it does suggest that finding the right set of semantic classes could be one big step towards improving system accuracy.

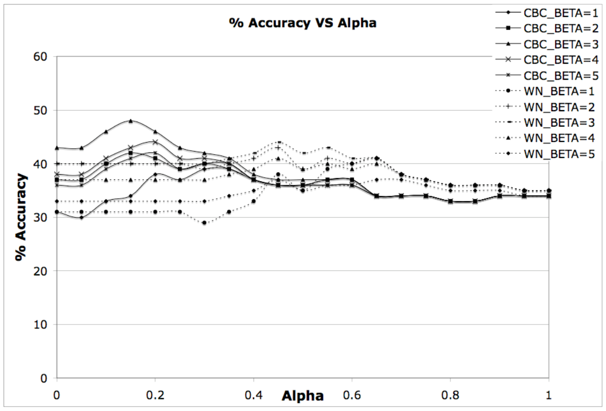


Figure 1: Accuracy variation for IRM with different values of α and β.

Two other factors that have a big impact on the performance of our systems are the values of the system parameters α and β, which decide the plausibility and directionality of an inference rule, respectively. To better study their effect on the system performances, we studied the two parameters independently.

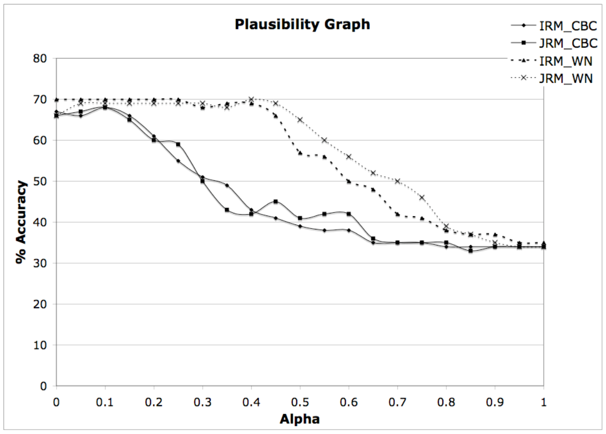


Figure 2: Accuracy variation in predicting correct versus incorrect inference rules for different values of α.

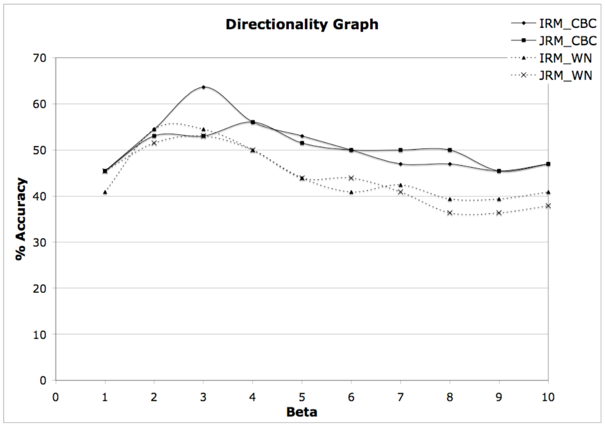


Figure 3: Accuracy variation in predicting directionality of correct inference rules for different values of β.

Figure 2 shows the variation in the accuracy for the task of predicting the correct and incorrect inference rules for the different systems when varying the value of α. To obtain this graph, we classified the inference rules in the test set only as correct and incorrect without further classification based on directionality. All of our four systems obtained accuracy scores in the range of 68-70% showing a good performance on the task of determining plausibility. This however is only a small improvement over the baseline score of 66% obtained by assuming every inference to be plausible (as will be shown below, our system has most impact not on determining plausibility but on determining directionality). Manual inspection of some system errors showed that the most common errors were due to the well-known 'problem of antonymy' when applying the distributional hypothesis. In DIRT, one can learn rules like "*X loves Y*" ⇔ "*X hates Y*". Since the plausibility of inference rules is determined by applying the distributional hypothesis and the antonym paths tend to take the same set of classes for X and Y, our models find it difficult to filter out the incorrect inference rules which DIRT ends up learning for this very same reason. To improve our system, one avenue of research is to focus specifically on filtering incorrect inference rules involving antonyms (perhaps using methods similar to (Lin et al. 2003)).

Figure 3 shows the variation in the accuracy for the task of predicting the directionality of the correct inference rules for the different systems when varying the value of β. To obtain this graph, we separated the correct inference rules form the incorrect ones and ran all the systems on only the correct ones, predicting only the directionality of each rule for different values of β. Too low a value of β means that the algorithms tend to predict most things as unidirectional and too high a value means that the algorithms tend to predict everything as bidirectional. It is clear from the figure that the performance of all the systems reach their peak performance in the range 2 ≤ β ≤ 4, which agrees with our intuition of obtaining the best system accuracy in a medium range. It is also seen that the best accuracy for each of the models goes up as compared to the corresponding values obtained in the general framework. The best performing system, IRM using CBC RSPs, reaches a peak accuracy of 63.64%, a much higher score than its accuracy score of 48% under the general framework and also a significant improvement over the baseline score of 48.48% for this task. Paired t-test shows that the difference is statistically significant at $p<0.05$. The baseline score for this task is obtained by assigning the most frequently occurring direction to all the correct inference rules. This paints a very encouraging picture about the ability of the algorithm to identify the directionality much more accurately if it can be provided with a cleaner set of inference rules.

6 Conclusion

Semantic inferences are fundamental to understanding natural language and are an integral part of many natural language applications such as question answering, summarization and textual entailment. Given the availability of large amounts of text and with the increase in computation power, learning them automatically from large text corpora has become increasingly feasible and popular. We introduced the Directionality Hypothesis, which states that if two paths share a significant number of relational selectional preferences (RSPs) and if the first has many more RSPs than the second, then the second path implies the first. Our experiments show empirical evidence that the Directionality Hypothesis with RSPs can indeed be used to filter incorrect inference rules and find the directionality of correct ones. We believe that this result is one step in the direction of solving the basic problem of semantic inference.

Several questions must still be addressed. The models need to be improved in order to address the problem of incorrect inference rules. The distributional hypothesis does not provide a framework to address the issue with antonymy relations like "*X loves Y*" ⇔ "*X hates Y*" and hence other ideas need to be investigated.

Ultimately, our goal is to improve the performance of NLP applications with better inferencing capabilities. Several recent data points, such as (Harabagiu and Hickl 2006), and others discussed in Section 2.1, give promise that refined inference rules for directionality may indeed improve question answering, textual entailment and multi-document summarization accuracies. It is our hope that methods such as the one proposed in this paper may one day be used to harness the richness of automatically created inference rule resources within large-scale NLP applications.

References

Anick, P.G. and Tipirneni, S. 1999. The Paraphrase Search Assistant: Terminology Feedback for Iterative Information Seeking. In *Proceedings of SIGIR 1999*. pp. 53-159. Berkeley, CA

Barzilay, R. and McKeown, K.R. 2001.Extracting Paraphrases from a Parallel Corpus. In *Proceedings of ACL 2001*. pp. 50–57. Toulose, France.

Barzilay, R.; McKeown, K.R. and Elhadad, M. 1999. Information Fusion in the Context of Multi-Document Summarization. In *Proceedings of ACL 1999*. College Park, Maryland.

Chklovski, T. and Pantel, P. 2004. VerbOCEAN: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of EMNLP 2004*. Barcellona Spain.

Cover, T.M. and Thomas, J.A. 1991. *Elements of Information Theory*. John Wiley & Sons.

Echihabi, A. and Marcu. D. 2003. A Noisy-Channel Approach to Question Answering. In *Proceedings of ACL 2003*. Sapporo, Japan.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Geffet, M.; Dagan, I. 2005. The Distributional Inclusion Hypothesis and Lexical Entailment. In *Proceedings of ACL 2005*. pp. 107-114. Ann Arbor, Michigan.

Harabagiu, S.; and Hickl, A. 2006. Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of ACL 2006*. pp. 905-912. Sydney, Australia.

Harris, Z. 1954. Distributional structure. *Word*. 10(23): 146-162.

Lenat, D. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Lin, D. 1993. Parsing Without OverGeneration. In *Proceedings of ACL 1993*. pp. 112-120. Columbus, OH.

Lin, D. and Pantel, P. 2001. Discovery of Inference Rules for Question Answering. *Natural Language Engineering* 7(4):343-360.

Lin, D.; Zhao, S.; Qin, L. and Zhou, M. 2003. Identifying Synonyms among Distributionally Similar Words. In *Proceedings of IJCAI 2003*, pp. 1492-1493. Acapulco, Mexico.

Manning, C.D. and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Moldovan, D.; Clark, C.; Harabagiu, S. and Maiorano S. 2003. COGEX: A Logic Prover for Question Answering. In *Proceedings of HLT/NAACL 2003*. Edmonton, Canada.

Pantel, P.; Bhagat, R.; Coppola, B.; Chklovski, T. and Hovy, E. 2007. ISP: Learning Inferential Selectional Preferences. In *Proceedings of HLT/NAACL 2007*. Rochester, NY.

Pantel, P. and Lin, D. 2002. Discovering Word Senses from Text. In *Proceedings of KDD 2002*. pp. 613-619. Edmonton, Canada.

Resnik, P. 1996. Selectional Constraints: An Information-Theoretic Model and its Computational Realization. *Cognition*, 61:127–159.

Siegel, S. and Castellan Jr., N. J. 1988. Nonparametric Statistics for the Behavioral Sciences. McGraw-Hill.

Szpektor, I.; Tanev, H.; Dagan, I.; and Coppola, B. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP 2004*. pp. 41-48. Barcelona, Spain.

Torisawa, K. 2006. Acquiring Inference Rules with Temporal Constraints by Using Japanese Coordinated Sentences and Noun-Verb Co-occurances. In *Proceedings of HLT/NAACL 2006*. pp. 57-64. New York, New York.

Wilks, Y. 1975. Preference Semantics. In E.L. Keenan (ed.), *Formal Semantics of Natural Language*. Cambridge: Cambridge University Press.

Zanzotto, F.M.; Pennacchiotti, M.; Pazienza, M.T. 2006. Discovering Asymmetric Entailment Relations between Verbs using Selectional Preferences. In *Proceedings of COLING/ACL 2006*. pp. 849-856. Sydney, Australia.

Modelling Polysemy in Adjective Classes by Multi-Label Classification

Gemma Boleda
GLiCom
Universitat Pompeu Fabra
08003 Barcelona
gemma.boleda@upf.edu

Sabine Schulte im Walde
IMS
University of Stuttgart
70174 Stuttgart
schulte@ims.uni-stuttgart.de

Toni Badia
GLiCom
Universitat Pompeu Fabra
08003 Barcelona
toni.badia@upf.edu

Abstract

This paper assesses the role of multi-label classification in modelling polysemy for language acquisition tasks. We focus on the acquisition of semantic classes for Catalan adjectives, and show that polysemy acquisition naturally suits architectures used for multi-label classification. Furthermore, we explore the performance of information drawn from different levels of linguistic description, using feature sets based on morphology, syntax, semantics, and n-gram distribution. Finally, we demonstrate that ensemble classifiers are a powerful and adequate way to combine different types of linguistic evidence: a simple, majority voting ensemble classifier improves the accuracy from 62.5% (best single classifier) to 84%.

1 Introduction

This paper reports on a series of experiments to explore the automatic acquisition of semantic classes for Catalan adjectives. The most important challenge of the classification task is to model the assignment of polysemous lexical instances to multiple semantic classes, combining a) a state-of-the-art Machine Learning architecture for *Multi-label Classification* (Schapire and Singer, 2000; Ghamrawi and McCallum, 2005) and an *Ensemble Classifier* (Dietterich, 2002) with b) the definition of features at various levels of linguistic description.

A proper treatment of polysemy is essential in the area of lexical acquisition, since polysemy represents a pervasive phenomenon in natural language. However, previous approaches to the automatic acquisition of semantic classes have mostly disregarded the problem (cf. Merlo and Stevenson, 2001 and Stevenson and Joanis, 2003 for English semantic verb classes, or Schulte im Walde, 2006 for German semantic verb classes). There are a few exceptions to this tradition, such as Pereira et al. (1993), Rooth et al. (1999), Korhonen et al. (2003), who used soft clustering methods for multiple assignment to verb semantic classes.

Our work addresses the lack of methodology in modelling a polysemous classification. We implement a multi-label classification architecture to handle polysemy. This paper concentrates on the classification of Catalan adjectives, but the general nature of the architecture should allow related tasks to profit from our insights.

As target classification for the experiments, a set of 210 Catalan adjectives was manually classified by experts into three simple and three polysemous semantic classes. We deliberately decided in favour of a small-scale, broad classification. So far, there is little work on the semantic classification of adjectives, as opposed to verbal semantic classification. The semantic classification we propose is a first step in characterising adjectival meaning, and can be refined and extended in subsequent work.

The experiments also provide a thorough comparison of feature sets based on different levels of linguistic description (morphology, syntax, semantics). A set of features is defined for each level of description, and its performance is assessed within the series of experiments. An ensemble classifier comple-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 171–180, Prague, June 2007. ©2007 Association for Computational Linguistics

ments the classification architecture, by optimising the combination of these different types of linguistic evidence.

Our task is motivated by the fact that adjectives play an important role in sentential semantics: they are crucial in determining the reference of NPs, and in defining properties of entities. Even using only three different classes, the information acquired could be applied to, e.g., identify referents in a given context in Dialog or Question Answering systems, and to induce properties of objects within Information Extraction tasks. Furthermore, with the semantic classes corresponding to broad sense representations, they can be exploited for Word Sense Disambiguation.

The remainder of this paper is organised as follows. Section 2 provides background on Catalan adjectives, and Section 3 presents the Gold Standard classification. Section 4 introduces the methodology of the multi-label classification experiments, Section 5 discusses the results, and the improved ensemble classifier is presented in Section 6.

2 Catalan adjective classes

The definition and characterisation of our target semantic classification follows the proposal by Raskin and Nirenburg (1998) within the framework of Ontological Semantics(Nirenburg and Raskin, 2004). In Ontological Semantics, an ontology of concepts modelling the world is explicitly defined, and the semantics of words are mapped onto elements of the ontology. The classification pursued in this paper is drawn up based on the ontological sort of adjectival denotation: all adjectives denote properties, but these properties can be instantiated as simple attributes (*basic adjectives*), relationships to objects (*object-related adjectives*), or relationships to events (*event-related adjectives*).

Basic adjectives are the prototypical adjectives which denote attributes or properties and cannot be decomposed further (such as *bonic* 'beautiful', *gran* 'big'). In Ontological Semantics, these adjectives are mapped to concepts of type *attribute*. For instance, the semantics of the adjective *gran* specifies a mapping to the *size-attribute* element in the ontology. As for event-related adjectives, they have an event component in their meaning and are therefore mapped onto *event* concepts in the ontology. For instance, the semantics of *tangible* ('tangible') includes a pointer to the event element *touch* in the ontology. Similarly, object-related adjectives are mapped onto object concepts in the ontology: *deformació nasal* ('nasal deformity') can be paraphrased as *deformity that affects the nose*, so *nasal* evokes the object *nose*.

The semantic distinctions are mirrored at several levels of linguistic description, such as morphology, syntax, and semantics. For instance, there is a clear relationship between morphological type and semantic class: basic adjectives are typically non-derived, object adjectives tend to be denominal, and event adjectives are usually deverbal. This is the default mapping that one expects from the morphology-semantics interface. As an example for syntactic evidence, basic adjectives in Catalan can be used non-restrictively (in a pre-nominal position) and also predicatively, while object adjectives typically cannot.

However, the correspondences between the linguistic properties and the semantic classes are not one-to-one mappings. Taking the morphological level as an example, some denominal adjectives are basic (such as *vergonyós* 'shy', from *vergonya* 'shyness'). Conversely, some object adjectives are not synchronically denominal (such as *botànic* 'botanical'), and some deverbal adjectives are not event-related, such as *amable* (lit. 'suitable to be loved'; has evolved to 'kind, friendly'). In such cases, the semantic class can be better traced in the distributional properties, not the morphological properties of the adjective.

The proposed classification accounts for some cases of adjectival polysemy. For instance, *familiar* has an object reading (related to the Catalan noun for 'family'), and a basic reading (corresponding to the English adjective 'familiar'):

(1) reunió familiar / cara familiar
meeting familiar / face familiar

'family meeting / familiar face'

Similarly, the participial adjective *sabut* ('known') has an event-related sense, corresponding to the verb *saber* ('know'), and a basic sense equivalent to 'wise':

(2) conseqüència sabuda / home sabut
consequence known / man wise
'known consequence / wise man'

The polysemy between our proposed classes, as exemplified in (1) and (2), is the kind of polysemy we aim to model in the acquisition experiments reported in this paper.

3 Gold Standard classes

As a Gold Standard for the experiments to follow, 210 Catalan adjectives were classified by three experts. The adjectives were randomly sampled from an adjective database (Sanromà, 2003), balancing three factors of variability: frequency, morphological type, and suffix. An equal number of adjectives was chosen from three frequency bands (low, medium, high), from four derivational types (denominal, deverbal, non-derived, participle), and from a series of suffixes within each type. The derivational type and suffix of each adjective were available in the adjective database, and had been manually encoded.

Three experts assigned the 210 lemmata to one out of six classes: each adjective was tagged as basic (B), event (E), object (O), or as polysemous between basic and event (BE), between basic and object (BO), or between event and object (EO). The decisions were reached by consensus. The distribution of the Gold Standard material across classes is shown in the last column of Table 6 (Section 5.2).

In the acquisition experiments, our aim is to automatically assign a class to each adjective that can be simple (B, E, O) or complex (BE, BO, EO), in case of polysemy.

4 Classification method

Adjective classification was performed within a two-level architecture for multi-label classification: first, make a binary decision on each of the classes, and then combine the classifications to achieve a final, multi-label classification. We therefore decomposed the global decision on the (possibly polysemous) class of an adjective into three binary decisions: Is it basic or not? Is it event-related or not? Is it object-related or not? The individual decisions were then combined into an overall classification that included polysemy. For example, if a lemma was classified both as basic and as object in each of the binary decisions, it was deemed polysemous (BO). The motivation behind this approach was that polysemous adjectives should exhibit properties of all the classes involved. As a result, positive decisions on each binary classification can be made by the algorithm, which can be viewed as implicit polysemous assignments.

This classification architecture is very popular in Machine Learning for multi-label problems, cf. (Schapire and Singer, 2000; Ghamrawi and McCallum, 2005), and has also been applied to NLP problems such as entity extraction and noun-phrase chunking (McDonald et al., 2005). The remainder of this section describes other methodological aspects of our experiments.

4.1 Classifier: Decision Trees

As classifier for the binary decisions we chose Decision Trees, one of the most widely used Machine Learning techniques for supervised experiments (Witten and Frank, 2005). Decision Trees provide a transparent representation of the decisions made by the algorithm, and thus facilitate the inspection of results and the error analysis. The experiments were carried out with the freely available Weka software package. The particular algorithm chosen, Weka's J48, is the latest open source version of C4.5 (Quinlan, 1993). For an explanation of decision tree induction and C4.5, see Quinlan (1993) and Witten and Frank (2005, Sections 4.3 and 6.1).

4.2 Feature definition

Five levels of linguistic description, formalised as different feature sets, were chosen for our task. They included evidence from morphology (*morph*), syntax (*func*, *uni*, *bi*), semantics (*sem*), plus a combination of the five levels (*all*). Table 1 lists the linguistic levels, their explanations, and the number of features used on each level.[1] Morphological features (*morph*) encode the derivational type (denominal, deverbal, participial, non-derived) and the suffix (in case the adjective is derived) of each adjective, and correspond to the manually encoded informa-

[1] In level *all*, different features were used for each of the three classes. Table 1 reports the mean number of features across the three classes.

Level	Explanation	# Features
morph	morphological (derivational) properties	2
func	syntactic function	4
uni	uni-gram distribution	24
bi	bi-gram distribution	50
sem	distributional cues of semantic properties	18
all	combination of the 5 linguistic levels	10.3

Table 1: Linguistic levels as feature sets.

tion from the adjective database. Syntactic and semantic features encode distributional properties of adjectives. Syntactic features comprise three subtypes: (i) the syntactic function (level *func*) of the adjective, as assigned by a shallow Constraint Grammar (Alsina et al., 2002), distinguishing the modifier (pre-nominal or post-nominal) and predicative functions; (ii) a unigram distribution (level *uni*), independently encoding the parts of speech (POS) of the words preceding and following the adjective, respectively; and (iii) a bigram distribution (level *bi*), the POS bigram around the target adjective, considering only the 50 most frequent bigrams to avoid sparse features. Semantic features (level *sem*) expand syntactic features with heterogeneous shallow cues of semantic properties. Table 2 lists the semantic properties encoded in the features, as well as the number of heuristic cues defined for each property. As an example, one of the shallow cues used for gradability was the presence of degree adverbs (*més* ‘more’, *menys* ‘less’) to the left of the target adjectives. The last set of features, *all*, combines features from all levels of description. However, it does not contain all features, but a selection of the most relevant ones (further details in Section 4.3).

property	#
non-restrictivity	1
predicativity	4
gradability	4
syntactic function of head noun	3
distance to the head noun	1
binaryhood (adjectives with two arguments)	1
agreement properties	2

Table 2: Semantic features.

4.3 Feature selection

Irrelevant features typically decrease performance by 5 to 10% when using Decision Trees (Witten and Frank, 2005, p. 288). We therefore applied a feature selection algorithm. We chose a feature selection method available in Weka (*WrapperSubsetEval*) that selects a subset of the features according to its performance within the Machine Learning algorithm used for classification. Accuracy for a given subset of features is estimated by cross-validation over the training data. Because the number of subsets increases exponentially with the number of features, this method is computationally very expensive, and we used a best-first search strategy to alleviate this problem.

We additionally used the feature selection procedure to select the features for level *all*: for each class, we used only those features that were selected by the feature selection algorithm in at least 30% of the experiments.

4.4 Differences across linguistic levels

One of our goals was to test the strengths and weaknesses of each level of linguistic description for the task of adjective classification. This was done by comparing the accuracy results obtained with each of the feature sets in the Machine Learning experiments. Following a standard procedure in Machine Learning, we created several partitions of the data to obtain different estimates of the accuracy of each of the levels, so as to be able to perform a significance test on the differences in accuracy. We performed 10 experiments with 10-fold cross-validation (*10x10 cv* for short), so that for each class 100 different binary decisions were made for each adjective. For the comparison of accuracies, a standard paired t-test could not be used, because of the inflated Type I error probability when reusing data (Dietterich, 1998). Instead, we used the *corrected resampled t-test* as proposed by Nadeau and Bengio (2003).[2]

5 Classification results

5.1 Accuracy results

The accuracy results for each of the binary decisions (basic/non-basic, event/non-event, object/non-object) are depicted in Table 3.[3] Level *bl* corresponds to the baseline: the baseline accuracy was determined by assigning all lemmata to the most frequent class. The remaining levels follow the nomenclature in Table 1 above. Each column contains the mean and the standard deviation (marked by ±) of the accuracy for the relevant level of information over the 100 results obtained with 10x10 cv.

	Basic	**Event**	**Object**
bl	65.2 ±11.1	76.2 ±9.9	71.9 ±9.6
morph	72.5 ±7.9	89.1 ±6.0	84.2 ±7.5
func	73.6 ±9.3	76.0 ±9.3	81.7 ±7.4
uni	66.1 ±9.4	75.1 ±10.6	82.2 ±7.5
bi	67.4 ±10.6	72.3 ±10.2	83.0 ±8.3
sem	72.8 ±9.0	73.8 ±9.6	82.3 ±8.0
all	**75.3** ±7.6	**89.4** ±5.7	**85.4** ±8.7

Table 3: Accuracy results for binary decisions.

As one might have expected, the best results were obtained with the *all* level (bold faced in Table 3), which is the combination of all feature types. This level achieved a mean improvement of 12.3% over the baseline. The differences in accuracy results between most levels of information were, however, rather small. For the object class, all levels except for *func* and *uni* achieved a significant improvement over the baseline. For the basic class, no improvement over the baseline was significant according to the corrected resampled t-test. And for the event class, only levels *morph* and *all* offered a significant improvement in accuracy; the remaining levels even obtained a slightly lower accuracy score.

These results concern the three individual binary decisions. However, our goal was not to obtain three separate decisions, but a single classification including polysemy. Table 4 shows the accuracy results for the classification obtained by combining the three individual decisions for each adjective. We report two accuracy measures, full and partial: full accuracy required the class assignments to be identical; partial accuracy only required some overlap in the classification of the Machine Learning algorithm and the Gold Standard for a given class assignment. The motivation for calculating partial overlap was that a class assignment with some overlap with the Gold Standard (even if they were not identical) is generally more useful than a class assignment with no overlap.

	Full	**Partial**
bl	51.0 ±0.0	65.2 ±0.0
morph	60.6 ±1.3	87.8 ±0.4
func	53.5 ±1.8	79.8 ±1.3
uni	52.3 ±1.7	76.7 ±1.0
bi	52.9 ±1.9	76.9 ±1.8
sem	52.0 ±1.3	78.7 ±1.7
all	**62.3** ±2.3	**90.7** ±1.6

Table 4: Accuracy results for combined decisions.

Again, the best results were obtained with the *all* level. The second best results were obtained with level *morph*. These results could have been expected from the results obtained by the individual decisions (Table 3); however, note that the differences between the various levels are much clearer in the combined classification than in the individual binary decisions.

Table 5 shows the two-by-two comparisons of the accuracy scores. Each cell contains the difference in accuracy means between two levels of description, as well as the level of significance of the difference. The significance is marked as follows: * for $p < 0.05$, ** for $p < 0.01$, *** for $p < 0.001$. If no asterisk is shown, the difference was not significant.

Under the strictest evaluation condition (full accu-

[2] Note that the corrected resampled t-test can only compare accuracies obtained under two conditions (algorithms or, as is our case, feature sets); ANOVA would be more adequate. In the field of Machine Learning, there is no established correction for ANOVA for the purposes of testing differences in accuracy (Bouckaert, 2004). Therefore, we used multiple t-tests instead, which increases the overall error probability of the results for the significance tests.

[3] The accuracy for each decision was computed independently. For instance, a *BE* adjective was judged correct within the basic class iff the decision was *basic*; correct within the event class iff the decision was *event*; and correct within the object class iff the decision was *non-object*.

agreement	level	bl	morph	func	uni	bi	sem
full	morph	9.7***					
	func	2.5*	-7.1***				
	uni	1.4	-8.3***	-1.1			
	bi	2.0	-7.7***	-0.6	0.6		
	sem	1.0	-8.7***	-1.5	-0.4	1.0	
	all	11.4***	1.7	8.9***	10.0***	9.4***	10.4***
partial	morph	-22.6***					
	func	14.6***	-8.0***				
	uni	11.4***	-11.1***	-3.1**			
	bi	11.7***	-10.9***	-2.9**	0.2		
	sem	13.4***	-9.1***	-1.1	2.0	1.8	
	all	25.4***	2.9*	10.9***	14.0***	13.8***	12.0***

Table 5: Comparison of accuracy scores across linguistic levels.

racy), only levels *morph*, *func*, and *all* significantly improved upon the baseline. Levels *morph* and *all* are better than the remaining levels, to a similar extent. In the partial evaluation condition, all levels achieved a highly significant improvement over the baseline ($p < 0.001$). Therefore, the classifications obtained with any of the feature levels are more useful than the baseline, in the sense that they present more overlap with the Gold Standard.

The best result obtained for the full classification of adjectives with our methodology achieved a mean of 62.3% (full accuracy) or 90.7% (partial accuracy), which represents an improvement of 11.3% and 25.5% over the baselines, respectively. Levels including morphological information were clearly superior to levels using only distributional information.

These results suggest that morphology is the best single source of evidence for our task. However, recall from Section 3 that the sampling procedure for the Gold Standard explicitly balanced for morphological factors. As a result, denominal and participial adjectives are underrepresented in the Gold Standard, while non-derived and deverbal adjectives are overrepresented. Moreover, previous experiments on different datasets (Boleda et al., 2004; Boleda et al., 2005) provided some evidence that distributional information outperforms morphological information for our task. Therefore, we cannot conclude from the experiments that morphological features are the most important information for the classification of Catalan adjectives in general.

5.2 Error analysis

The error analysis focuses on the two best feature sets, *morph* and *all*. Table 6 compares the errors made by the experiment classifications (based on the two sets of features) against the Gold Standard classification. To obtain a unique experiment classification for each feature level in this comparison, we applied majority voting across the 10 different classifications obtained in the 10 experiment runs for each of the linguistic levels. The table rows correspond to the Gold Standard classification and the columns correspond to the experiment classifications with the feature levels *all* and *morph*, respectively. The matches (the diagonal elements) are in italics, and off-diagonal cells representing the largest numbers of mismatches are boldfaced. The overall number of mistakes made by both levels with majority voting is almost the same: 86 (*morph*) vs. 89 (*all*). However, the mismatches are qualitatively quite different.

Level *morph* uniformly mapped denominal adjectives to both basic and object (BO). Because of this overgeneration of BOs, 31 lemmata that were tagged as either basic or object in the Gold Standard were assigned to BO. In contrast, level *all* was overly discriminative: most of the BO cases (16 out of 23), as well as 16 object adjectives, were assigned to basic. This type of confusion could be explained by the fact that some non-prototypical basic adjectives were as-

		all						**morph**						
		B	BE	BO	E	EO	O	B	BE	BO	E	EO	O	*Total*
GS	B	*94*	**12**	0	0	1	0	*82*	2	**10**	**11**	2	0	*107*
	BE	1	*6*	0	0	0	0	0	*1*	0	6	0	0	*7*
	BO	**16**	1	*5*	1	0	0	5	0	*16*	2	0	0	*23*
	E	5	**23**	1	*7*	1	0	4	7	0	*25*	1	0	*37*
	EO	0	2	0	0	*4*	0	0	0	0	6	*0*	0	*6*
	O	**16**	1	6	2	0	*5*	6	0	**21**	3	0	*0*	*30*
	Total	*132*	*45*	*12*	*10*	*6*	*5*	*97*	*10*	*47*	*53*	*3*	*0*	*210*

Table 6: Levels *all* and *morph* against the Gold Standard.

signed to the basic class in the Gold Standard, because they did not fit the narrower definitions of the event and object classes, but these adjectives do not behave like typical basic adjectives.

As for event adjectives, the *morph* level assigned almost all deverbal adjectives to the event class, which worked well in most cases (26). However, this mapping cannot distinguish deverbal adjectives with a basic meaning (11 basic and 6 BE adjectives in the Gold Standard). Level *all*, including morphological and distributional cues, also shows difficulties with the event class, but of a different nature. Feature examination showed that the distributional differences between basic and event adjectives are not robust. For instance, according to t-tests performed on the Gold Standard (α = 0.05), only three of the 18 semantic features exhibit significant mean differences for classes basic and event. In contrast, ANOVA across the 6 classes (α = 0.05) yields significant differences for 16 out of the 18 features, which indicates that most features serve to distinguish object adjectives from basic and event adjectives. As a result of the lack of robust distributional differences between basic and event adjectives, 35 basic or event adjectives were classified as BE when using the *all* level as feature set.

Further 23 event adjectives were incorrectly classified as BE by the *all* level, but correctly classified by the *morph* level, because they are deverbal adjectives. These cases involved adjectives derived from stative verbs, such as *abundant* ('abundant') or *preferible* ('preferable'). Feature analysis revealed that deverbal adjectives derived from stative verbs are more similar to basic adjectives than those derived from process-denoting verbs.

To sum up, the default morphological mapping mentioned in Section 2 works well in most cases but has a clear ceiling, as it cannot account for deviations from the expected mapping. Distributional cues are more sensitive to these deviations, but fail mostly in the distinction between basic and event, because the differences in syntactic distribution between these classes are not robust.

6 An improved classifier

The error analysis in the previous section has shown that, although the number of mistakes made with level *morph* and *all* is comparable, the kinds of mistakes are qualitatively very different. This suggests that mixing features for the construction of a single Decision Tree, as is done in level *all*, is not the optimal way to combine the strengths of each level of description. An alternative combination can be achieved with an *ensemble classifier*, a type of classifier that has received much attention in the Machine Learning community in the last decade (Dietterich, 2002). When building an ensemble classifier, several class proposals for each item are obtained, and one of them is chosen on the basis of majority voting, weighted voting, or more sophisticated decision methods. It has been shown that in most cases, the accuracy of the ensemble classifier is higher than the best individual classifier (Freund and Schapire, 1996; Dietterich, 2000; Breiman, 2001). Within NLP, ensemble classifiers have been applied, for instance, to genus term disambiguation in machine-readable dictionaries (Rigau et al., 1997), using a majority voting scheme upon several heuristics, and to part of speech tagging, by combining the class predictions of different algorithms (van Halteren et

Levels	Full Ac.	Part. Ac.
morph+func+uni+bi+sem+all	84.0 ±0.06	95.7 ±0.02
func+uni+bi+sem	81.5 ±0.04	95.9 ±0.01
morph+func+sem+all	72.4 ±0.03	89.3 ±0.02
bl	51.0 ±0.0	65.2 ±0.0
all	62.3 ±2.3	90.7 ±1.6

Table 7: Results for ensemble classifier.

al., 1998). The main reason for the general success of ensemble classifiers is that they gloss over the biases introduced by the individual systems.

We implemented an ensemble classifier by using the different levels of description as different subsets of features, and applying majority voting across the class proposals from each level. Intuitively, this architecture is analogous to having a team of linguists and NLP engineers, each contributing their knowledge on morphology, n-gram distribution, syntactic properties, etc., and have them reach a consensus classification. We thus established a different classification for each of the 10 cross-validation runs by assigning each adjective to the class that received most votes. To enable a majority vote, at least three levels have to be combined. Table 7 contains a representative selection of the combinations, together with their accuracies. Also, the accuracies obtained with the baseline (*bl*) and the best single level (*all*) are included for comparison.

In any of the combinations tested, accuracy improved over 10% with respect to the *all* level. The best result, a mean of 84% (full accuracy), was obtained by combining all levels of description. These results represent a raw improvement over the baseline of 33%, and 21.7% over the best single classifier. Also note that with this procedure 95.7% of the classifications obtained with the ensemble classifier present partial overlap with the class assignments in the Gold Standard.

These results show that the combination of different sources of linguistic evidence is more important than the type of information used. As an example, consider the second ensemble classifier in Table 7: this classifier excludes the two levels that contain morphological information (*morph* and *all*), which represents the most successful individual source of information for our dataset. Nevertheless, the combination achieved 19.2/20.9% more accuracy than levels *all* and *morph*, respectively.

7 Related work

Adjectives have received less attention than verbs and nouns within Lexical Acquisition. Work by Hatzivassiloglou and colleagues (Hatzivassiloglou and McKeown, 1993; Hatzivassiloglou and McKeown, 1997; Hatzivassiloglou and Wiebe, 2000) used clustering methods to automatically identify adjectival scales from corpora.

Coordination information was used in Bohnet et al. (2002) for a classification task similar to the task we pursue, using a bootstrapping approach. The authors, however, pursued a classification that is not purely semantic, between quantitative adjectives (similar to determiners, like *viele* 'many'), referential adjectives (*heutige*, 'of today'), qualitative adjectives (equivalent to basic adjectives), classificatory adjectives (equivalent to object adjectives), and adjectives of origin (*Stuttgarter*, 'from Stuttgart').

In a recent paper, Yallop et al. (2005) reported experiments on the acquisition of syntactic subcategorisation patterns for English adjectives.

Apart from the above research with a classificatory flavour, other lines of research exploited lexical relations among adjectives for Word Sense Disambiguation (Justeson and Katz, 1995; Chao and Dyer, 2000). Work by Lapata (2001), contrary to the studies mentioned so far, focused on the meaning of adjective-noun combinations, not on that of adjectives alone.

8 Conclusion

This paper has presented an architecture for the semantic classification of Catalan adjectives that explicitly includes polysemous classes. The focus of the architecture was on two issues: *(i) finding an appropriate set of linguistic features,* and *(ii) defining an adequate architecture for the task.* The investigation and comparison of features at various linguistic levels has shown that morphology plays a major role for the target classification, despite the caveats raised in the discussion. Morphological features related to derivational processes are among the simplest types of features to extract, so that the approach can be straightforwardly extended to languages similar to Catalan with no extensive need of resources.

Furthermore, we have argued that polysemy acquisition naturally suits multi-label classification architectures. We have implemented a standard architecture for this class of problems, and demonstrated its applicability and success. The general nature of the architecture should be useful for related tasks that involve polysemy within the area of automatic lexical acquisition.

Our work has focused on a broad classification of the adjectives, similarly to Merlo and Stevenson (2001), who classified transitive English verbs into three semantic classes. The small number of classes might be considered as an over-simplification of adjective semantics, but the simplified setup facilitates a detailed qualitative evaluation. In addition, as there has been virtually no work on the acquisition of semantic classes for adjectives, it seems sensible to start with a small number of classes and incrementally build upon that. Previous work has demonstrated that multi-label classification is applicable also to a large number of classes as used in, e.g., document categorisation (Schapire and Singer, 2000). This potential can be exploited in future work, addressing a finer-grained adjective classification.

Finally, we have demonstrated that the combination of different types of linguistic evidence boosts the performance of the system beyond the best single type of information: ensemble classifiers are a more adequate way to combine the linguistic levels of description than simply merging all features for tree construction. Using a simple, majority voting ensemble classifier, the accuracy jumped from 62.5% (best single classifier) to 84%. This result is impressive by itself, and also in comparison to similar work such as (Rigau et al., 1997), who achieved a 9% improvement on a similar task. Our insights are therefore useful in related work which involves the selection of linguistic features in Machine Learning experiments.

Future work involves three main lines of research. First, the refinement of the classification itself, based on the results of the experiments presented. Second, the use of additional linguistic evidence that contributes to the semantic class distinctions (e.g., selectional restrictions). Third, the application of the acquired information to broader NLP tasks. For example, given that each semantic class exhibits a particular syntactic behaviour, information on the semantic class should improve POS-tagging for adjective-noun and adjective-participle ambiguities, probably the most difficult distinctions both for humans and computers (Marcus et al., 1993; Brants, 2000). Also, semantic classes might be useful in terminology extraction, where, presumably, object adjectives participate in terms more often than basic adjectives.[4]

Acknowledgements

We thank Roser Sanromà for providing us with the manually annotated database of adjectives, and for being part of the Gold Standard annotation committee. Special thanks also to the Institut d'Estudis Catalans for proding us with the research corpus. The comments of three anonymous reviewers helped improve the paper. Finally, the financial support of the Universitat Pompeu Fabra and its Translation and Philology Department and the Fundación Caja Madrid is gratefully acknowledged.

References

À. Alsina, T. Badia, G. Boleda, S. Bott, À. Gil, M. Quixal, and O. Valentín. 2002. CATCG: a general purpose parsing tool applied. In *Proceedings of Third International Conference on Language Resources and Evaluation (LREC-02)*, Las Palmas, Spain.

B. Bohnet, S. Klatt, and L. Wanner. 2002. An approach to automatic annotation of functional information to adjectives with an application to German. In *Proceedings of the LREC Workshop on Linguistic Knowledge Acquisition and Representation*, Las Palmas, Spain.

G. Boleda, T. Badia, and E. Batlle. 2004. Acquisition of semantic classes for adjectives from distributional evidence. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 1119–1125, Geneva, Switzerland.

G. Boleda, T. Badia, and S. Schulte im Walde. 2005. Morphology vs. syntax in adjective class acquisition. In *Proceedings of the ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition*, pages 1119–1125, Ann Arbor, USA.

R. Bouckaert. 2004. Estimating replicability of classifier learning experiments. In *Proceedings of ICML*.

T. Brants. 2000. Inter-annotator agreement for a german newspaper corpus. In *Second International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

[4] Horacio Rodríguez, p. c., April 2007.

L. Breiman. 2001. Random forests. *Mach. Learn.*, 45:5–23.

G. Chao and M. G. Dyer. 2000. Word sense disambiguation of adjectives using probabilistic networks. In *Proceedings of COLING*, pages 152–158.

T.G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924.

T.G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, 40:5–23.

T.G. Dietterich. 2002. Ensemble learning. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. The MIT Press.

Y. Freund and R.E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proceedings of ICML*, pages 148–156.

N. Ghamrawi and A. McCallum. 2005. Collective multilabel classification. In *Proceedings of 14th Conf. on Information and Knowledge Management*.

V. Hatzivassiloglou and K. R. McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of ACL*, pages 172–182.

V. Hatzivassiloglou and K.R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of ACL/EACL*, pages 174–181.

V. Hatzivassiloglou and J. M. Wiebe. 2000. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of COLING*, pages 299–305, Morristown, NJ, USA. Association for Computational Linguistics.

J. S. Justeson and S. M. Katz. 1995. Principled disambiguation: Discriminating adjective senses with modified nouns. *Computational Linguistics*, 21(1):1–27.

A. Korhonen, Y. Krymolowski, and Z. Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of ACL*, pages 64–71.

M. Lapata. 2001. A corpus-based account of regular polysemy: The case of context-sensitive adjectives. In *Proceedings of NAACL*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

R. McDonald, K. Crammer, and F. Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *Proceedings of HLT-EMNLP*, pages 987–994.

P. Merlo and S. Stevenson. 2001. Automatic verb classification based on statistical distributions of argument structure. *Comp. Ling.*, 27(3):373–408.

C. Nadeau and Y. Bengio. 2003. Inference for the generalization error. *Mach. Learn.*, 52(3):239–281.

S. Nirenburg and V. Raskin. 2004. *Ontological Semantics*. MIT Press.

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional Clustering of English Words. In *Proceedings of ACL*, pages 183–190.

R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

V. Raskin and S. Nirenburg. 1998. An applied ontological semantic microtheory of adjective meaning for natural language processing. *Mach. Trans.*, 13(2-3):135–227.

G. Rigau, J. Atserias, and E. Agirre. 1997. Combining unsupervised lexical knowledge methods for word sense disambiguation. In *Proceedings of EACL*, pages 48–55.

M. Rooth, S. Riezler, D. Prescher, G. Carroll, and F. Beil. 1999. Inducing a Semantically Annotated Lexicon via EM-Based Clustering. In *Proceedings of ACL*.

R. Sanromà. 2003. Aspectes morfològics i sintàctics dels adjectius en català. Master's thesis, Universitat Pompeu Fabra.

R.E. Schapire and Y. Singer. 2000. Boostexter: A boosting-based system for text categorization. *Mach. Learn.*, 39(2-3):135–168.

S. Stevenson and E. Joanis. 2003. Semi-supervised verb class discovery using noisy features. In *Proceedings of CoNLL*.

H. van Halteren, J. Zavrel, and W. Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of ACL*, pages 491–497.

I.H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

J. Yallop, A. Korhonen, and T. Briscoe. 2005. Automatic acquisition of adjectival subcategorization from corpora. In *Proceedings of ACL*, Ann Arbor, Michigan.

Improving Query Spelling Correction Using Web Search Results

Qing Chen
Natural Language Processing Lab
Northeastern University
Shenyang, Liaoning, China, 110004
chenqing@ics.neu.edu.cn

Mu Li
Microsoft Research Asia
5F Sigma Center
Zhichun Road, Haidian District
Beijing, China, 100080
muli@microsoft.com

Ming Zhou
Microsoft Research Asia
5F Sigma Center
Zhichun Road, Haidian District
Beijing, China, 100080
mingzhou@microsoft.com

Abstract

Traditional research on spelling correction in natural language processing and information retrieval literature mostly relies on pre-defined lexicons to detect spelling errors. But this method does not work well for web query spelling correction, because there is no lexicon that can cover the vast amount of terms occurring across the web. Recent work showed that using search query logs helps to solve this problem to some extent. However, such approaches cannot deal with rarely-used query terms well due to the data sparseness problem. In this paper, a novel method is proposed for use of web search results to improve the existing query spelling correction models solely based on query logs by leveraging the rich information on the web related to the query and its top-ranked candidate. Experiments are performed based on real-world queries randomly sampled from search engine's daily logs, and the results show that our new method can achieve 16.9% relative *F*-measure improvement and 35.4% overall error rate reduction in comparison with the baseline method.

1 Introduction

Nowadays more and more people are using Internet search engine to locate information on the web. Search engines take text queries that users type as input, and present users with information of ranked web pages related to users' queries. During this process, one of the important factors that lead to poor search results is misspelled query terms. Actually misspelled queries are rather commonly observed in query logs, as shown in previous investigations into the search engine's log data that around 10%~15% queries were misspelled (Cucerzan and Brill, 2004).

Sometimes misspellings are due to simple typographic errors such as *teh* for *the*. In many cases the spelling errors are more complicated cognitive errors such as *camoflauge* for *camouflage*. As a matter of fact, correct spelling is not always an easy task – even many Americans cannot exactly spell out California governor's last name: *Schwarzenegger*. A spelling correction tool can help improve users' efficiency in the first case, but it is more useful in the latter since the users cannot figure out the correct spelling by themselves.

There has been a long history of general-purpose spelling correction research in natural language processing and information retrieval literature (Kukich, 1992), but its application to web search

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 181–189, Prague, June 2007.

query is still a new challenge. Although there are some similarities in correction candidate generation and selection, these two settings are quite different in one fundamental problem: How to determine the validity of a search term. Traditionally, the measure is mostly based on a pre-defined spelling lexicon – all character strings that cannot be found in the lexicon are judged to be invalid. However, in the web search context, there is little hope that we can construct such a lexicon with ideal coverage of web search terms. For example, even manually collecting a full list of car names and company names will be a formidable task.

To obtain more accurate understanding of this problem, we performed a detailed investigation over one week's MSN daily query logs, among which found that 16.5% of search terms are out of the scope of our spelling lexicon containing around 200,000 entries. In order to get more specific numbers, we also manually labeled a query data set that contains 2,323 randomly sampled queries and 6,318 terms. In this data set, the ratio of out-of-vocabulary (OOV) terms is 17.4%, which is very similar to the overall distribution. However, only 25.3% of these OOV terms are identified to be misspelled, which occupy 85% of the overall spelling errors. All these statistics indicate that accurate OOV term classification is of crucial importance to good query spelling correction performance.

Cucerzan and Brill (2004) first investigated this issue and proposed to use query logs to infer correct spellings of misspelled terms. Their principle can be summarized as follows: given an input query string *q*, finding a more probable query *c* than *q* within a confusion set of *q*, in which the edit distance between each element and *q* is less than a given threshold. They reported good recall for misspelled terms, but without detailed discussions on accurate classification of valid out-of-vocabulary terms and misspellings. In Li's work, distributional similarity metrics estimated from query logs were proposed to be used to discriminate high-frequent spelling errors such as *massenger* from valid out-of-vocabulary terms such as *biocycle*. But this method suffers from the data sparseness problem: sufficient amounts of occurrences of every possible misspelling and valid terms are required to make good estimation of distributional similarity metrics; thus this method does not work well for rarely-used out-of-vocabulary search terms and uncommon misspellings.

In this paper we propose to use web search results to further improve the performance of query spelling correction models. The key contribution of our work is to identify that the dynamic online search results can serve as additional evidence to determine users' intended spelling of a given term. The information in web search results we used includes the number of pages matched for the query, the term distribution in the web page snippets and URLs. We studied two schemes to make use of the returning results of a web search engine. The first one only exploits indicators of the input query's returning results, while the other also looks at other potential correction candidate's search results. We performed extensive evaluations on a query set randomly sampled from search engines' daily query logs, and experimental results show that we can achieve 35.4% overall error rate reduction and 18.2% relative *F*-measure improvement on OOV misspelled terms.

The rest of the paper is structured as follows. Section 2 details other related work of spelling correction research. In section 3, we show the intuitive motivations to use web search results for the query spelling correction. After presenting the formal statement of the query spelling correction problem in Section 4, we describe our approaches that use machine learning methods to integrate statistical features from web search results in Section 5. We present our evaluation methods for the proposed methods and analyze their performance in Section 6. Section 7 concludes the paper.

2 Related Work

Spelling correction models in most previous work were constructed based on conventional task settings. Based on the focus of these task settings, two lines of research have been applied to deal with non-word errors and real-word errors respectively.

Non-word error spelling correction is focused on the task of generating and ranking a list of possible spelling corrections for each word not existing in a spelling lexicon. Traditionally candidate ranking is based on manually tuned scores such as assigning alternative weights to different edit operations or leveraging candidate frequencies (Damerau, 1964; Levenshtein, 1966). In recent years, statistical models have been widely used for the tasks of natural language processing, including spelling correction task. (Brill and Moore, 2000) presented an improved error model over the one proposed by (Kernighan et al., 1990) by allowing generic string-to-string edit operations, which helps with modeling major cognitive errors such as the confusion between *le* and *al*. Via explicit modeling of phonetic information of English words, (Toutanova and Moore, 2002) further investigated this issue. Both of them require misspelled/correct word pairs for training, and the latter also needs a pronunciation lexicon, but recently (Ahmad and Kondrak, 2005) demonstrated that it is also possible to learn such models automatically from query logs with the EM algorithm, which is similar to work of (Martin, 2004), learning from a very large corpus of raw text for removing non-word spelling errors in large corpus. All the work for non-word spelling correction focused on the current word itself without taking into account contextual information.

Real-word spelling correction is also referred to be context sensitive spelling correction (CSSC), which tries to detect incorrect usage of valid words in certain contexts. Using a pre-defined confusion set is a common strategy for this task, such as in the work of (Golding and Roth, 1996) and (Mangu and Brill, 1997). Opposite to non-word spelling correction, in this direction only contextual evidences were taken into account for modeling by assuming all spelling similarities are equal.

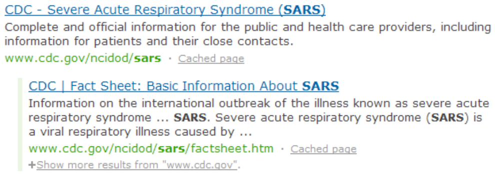
The complexity of query spelling correction task requires the combination of these types of evidence, as done in (Cucerzan and Brill, 2004; Li et al., 2006). One important contribution of our work is that we use web search results as extended contextual information beyond query strings by taking advantage of application specific knowledge. Although the information used in our methods can all be accessed in a search engine's web archive, such a strategy involves web-scale data processing which is a big engineering challenge, while our method is a light-weight solution to this issue.

3 Motivation

When a spelling correction model tries to make a decision whether to make a suggestion *c* to a query *q*, it generally needs to leverage two types of evidence: the similarity between *c* and *q*, and the validity plausibility of *c* and *q*. All the previous work estimated plausibility of a query based on the query string itself – typically it is represented as the string probability, which is further decomposed into production of consecutive n-gram probabilities. For example, both the work of (Cucerzan and Brill, 2004; Li et al., 2006) used n-gram statistical language models trained from search engine's query logs to estimate the query string probability.

In the following, we will show that the search results for a query can serve as a feedback mechanism to provide additional evidences to make better spelling correction decisions. The usefulness of web search results can be two-fold:

First, search results can be used to validate query terms, especially those not popular enough in query logs. One case is the validation for navigational queries (Broder, 2004). Navigational queries usually contain terms that are key parts of destination URLs, which may be out-of-vocabulary terms since there are millions of sites on the web. Because some of these navigational terms are very relatively rare in query logs, without knowledge of the special navigational property of a term, a query spelling correction model might confuse them with other low-frequency misspellings. But such information can be effectively obtained from the URLs of retrieved web pages. Inferring navigational queries through term-URL matching thus can help reduce the chance that the spelling correction model changes an uncommon web site name into popular search term, such as from *innovet* to *innovate*. Another example is that search results can be used in identifying acronyms or other abbreviations. We can observe some clear text patterns that relate abbreviations to their full spellings in the search results as shown in Figure 1. But such mappings cannot easily be obtained from query logs.

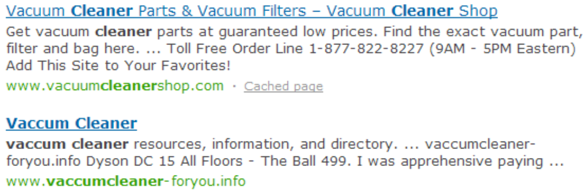


CDC - Severe Acute Respiratory Syndrome (**SARS**)
Complete and official information for the public and health care providers, including information for patients and their close contacts.
www.cdc.gov/ncidod/**sars** · Cached page

CDC | Fact Sheet: Basic Information About **SARS**
Information on the international outbreak of the illness known as severe acute respiratory syndrome ... **SARS**. Severe acute respiratory syndrome (**SARS**) is a viral respiratory illness caused by ...
www.cdc.gov/ncidod/**sars**/factsheet.htm · Cached page
+Show more results from "www.cdc.gov".

Figure 1. Sample search results for *SARS*

Second, search results can help verify correction candidates. The terms appearing in search results, both in the web page titles and snippets, provide additional evidences for users intention. For example, if a user searches for a misspelled query *vaccum cleaner* on a search engine, it is very likely that he will obtain some search results containing the correct term *vacuum* as shown in Figure 2. This can be attributed to the collective link text distribution on the web – many links with misspelled text point to sites with correct spellings. Such evidences can boost the confidence of a spelling correction model to suggest *vacuum* as a correction.



Vacuum **Cleaner** Parts & Vacuum Filters – Vacuum **Cleaner** Shop
Get vacuum **cleaner** parts at guaranteed low prices. Find the exact vacuum part, filter and bag here. ... Toll Free Order Line 1-877-822-8227 (9AM - 5PM Eastern) Add This Site to Your Favorites!
www.vacuum**cleaner**shop.com · Cached page

Vaccum Cleaner
vaccum cleaner resources, information, and directory. ... vaccumcleaner-foryou.info Dyson DC 15 All Floors - The Ball 499. I was apprehensive paying ...
www.**vaccumcleaner**-foryou.info

Figure 2. Sample search results for *vaccum cleaner*

The number of matched pages can be used to measure the popularity of a query on the web, which is similar to term frequencies occurring in query logs, but with broader coverage. Poor correction candidates can usually be verified by a smaller number of matched web pages.

Another observation is that the documents retrieved with correctly-spelled query and misspelled ones are similar to some extent in the view of term distribution. Both the web retrieval results of *vacuum* and *vaccum* contain terms such as *cleaner*, *pump*, *bag* or *systems*. We can take this similarity as an evidence to verify the spelling correction results.

4 Problem Statement

Given a query q, a spelling correction model is to find a query string c that maximizes the posterior probability of c given q within the confusion set of q. Formally we can write this as follows:

$$c^* = \underset{c \in C}{\textbf{argmax}}\; Pr(c|q) \tag{1}$$

where C is the confusion set of q. Each query string c in the confusion set is a correction candidate for q, which satisfies the constraint that the spelling similarity between c and q is within given threshold δ.

In this formulation, the error detection and correction are performed in a unified way. The query q itself always belongs to its confusion set C, and when the spelling correction model identifies a more probable query string c in C which is different from q, it claims a spelling error detected and makes a correction suggestion c.

There are two tasks in this framework. One is how to learn a statistical model to estimate the conditional probability $Pr(c|q)$, and the other is how to generate confusion set C of a given query q.

4.1 Maximum Entropy Model for Query Spelling Correction

We take a feature-based approach to model the posterior probability $Pr(c|q)$. Specifically we use the maximum entropy model (Berger et al., 1996) for this task:

$$Pr(c|q) = \frac{\exp(\sum_{i=1}^{N} \lambda_i f_i(c,q))}{\sum_c \exp(\sum_{i=1}^{N} \lambda_i f_i(c,q))} \tag{2}$$

where $\sum_c \exp(\sum_{i=1}^{N} \lambda_i f_i(c,q))$ is the normalization factor; $f_i(c,q)$ is a feature function defined over query q and correction candidate c , while λ_i is the corresponding feature weight. λs can be optimized using the numerical optimization algorithms such as the Generalized Iterative Scaling (GIS) algorithm (Darroch and Ratcliff, 1972) by maximizing the posterior probability of the training set which contains a manually labeled set of query-truth pairs:

$$\lambda^* = \operatorname{argmax} \sum_{c,q} \log Pr_\lambda(c|q) \tag{3}$$

The advantage of maximum entropy model is that it provides a natural way and unified framework to integrate all available information sources. This property is well fit for our task in which we are using a wide variety of evidences based on lexicon, query log and web search results.

4.2 Correction Candidate Generation

Correction candidate generation for a query q can be decomposed into two phases. In the first phase, correction candidates are generated for each term in the query from a term-base extracted from query logs. This task can leverage conventional spelling correction methods such as generating candidates based on edit distance (Cucerzan and Brill, 2004) or phonetic similarity (Philips, 1990). Then the correction candidates of the entire query are generated by composing the correction candidates of each individual term. Let $q = w_1 \cdots w_n$, and the confusion set of w_i is C_{w_i}, then the confusion set of q is $C_{w_1} \otimes C_{w_2} \otimes \cdots \otimes C_{w_n}$[1]. For example, for a query $q = w_1 w_2$, w_1 has candidates c_{11} and c_{12}, while w_2 has candidates c_{21} and c_{22}, then the confusion set C is $\{c_{11}c_{21}, c_{11}c_{22}, c_{12}c_{21}, c_{12}c_{22}\}$.

[1] For denotation simplicity, we do not cover compound and composition errors here.

The problem of this method is the size of confusion set C may be huge for multi-term queries. In practice, one term may have hundreds of possible candidates, then a query containing several terms may have millions. This might lead to impractical search and training using the maximum entropy modeling method. Our solution to this problem is to use *candidate pruning*. We first roughly rank the candidates based on the statistical n-gram language model estimated from query logs. Then we only choose a subset of C that contains a specified number of top-ranked (most probable) candidates to present to the maximum entropy model for offline training and online re-ranking, and the number of candidates is used as a parameter to balance top-line performance and run-time efficiency. This subset can be efficiently generated as shown in (Li et al., 2006).

5 Web Search Results based Query Spelling Correction

In this section we will describe in detail the methods for use of web search results in the query spelling correction task. In our work we studied two schemes. The first one only employs indicators of the input query's search results, while the other also looks at the most probable correction candidates' search results. For each scheme, we extract additional scheme-specific features from the available search results, combine them with baseline features and construct a new maximal model to perform candidate ranking.

5.1 Baseline model

We denote the maximum entropy model based on baseline model feature set as M0 and the feature set S0 derived from the latest state of the art works of (Li et al., 2006), where S0 includes the features mostly concerning the statistics of the query terms and the similarities between query terms and their correction candidates.

5.2 Scheme 1: Using search results for input query only

In this scheme we build more features for each correction candidate (including input query q itself) by distilling more evidence from the search results of the query. S1 denotes the augmented feature set, and M1 denotes the maximum entropy model based on S1. The features are listed as follows:

1. **Number of pages returned**: the number of web search pages retrieved by a web search engine, which is used to estimate the popularity of query. This feature is only for q.
2. **URL string**: Binary features indicating whether the combination of terms of each candidate is in the URLs of top retrieved documents. This feature is for all candidates.
3. **Frequency of correction candidate term**: the number of occurrences of modified terms in the correction candidate found in the title and snippet of top retrieved documents based on the observation that correction terms possibly co-occur with their misspelled ones. This feature is invalid for q.
4. **Frequency of query term**: the number of occurrences of each term of q found in the title or snippet of the top retrieved documents, based on the observation that the correct terms always appear frequently in their search results.
5. **Abbreviation pattern**: Binary features indicating whether inputted query terms might be abbreviations according to text patterns in search results.

5.3 Scheme 2: Using both search results of input query and top-ranked candidate

In this scheme we extend the use of search results both for query q and for top-ranked candidate c other than q determined by M1. First we submit a query to a search engine for the initial retrieval to obtain one set of search results R_q, then use M1 to find the best correction candidate c other than q. Next we perform a second retrieval with c to obtain another set of search results R_c. Finally additional features are generated for each candidate based on R_c, then a new maximum entropy model M2 is built to re-rank the candidates for a second time. The entire process can be schematically shown in Figure 3.

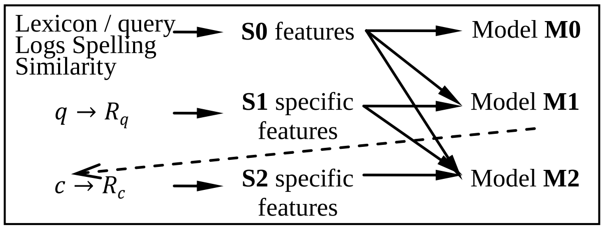


Figure 3. Relations of models and features

where R_q is the web search results of query q; R_c is the web search results of c which is the top-ranked correction of q suggested by model M1.

The new feature set denoted with S2 is a set of document similarities between R_q and R_c, which includes different similarity estimations between the query and its correction at the document level using merely cosine measure based on term frequency vectors of R_q and R_c.

6 Experiments

6.1 Evaluation Metrics

In our work, we consider the following four types of evaluation metrics:

- **Accuracy**: The number of correct outputs proposed by the spelling correction model divided by the total number of queries in the test set
- **Recall**: The number of correct suggestions for misspelled queries by the spelling correction model divided by the total number of misspelled queries in the test set
- **Precision**: The number of correct suggestions for misspelled queries proposed by the spelling correction model divided by the total number of suggestions made by the system
- **F-measure**: Formula $F = 2PR/(P + R)$ used for calculating the f-measure, which is essentially the harmonic mean of recall and precision

Any individual metric above might not be sufficient to indicate the overall performance of a query spelling correction model. For example, as in most retrieval tasks, we can trade recall for precision or vice versa. Although intuitively F might be in accordance with accuracy, there is no strict theoretical relation between these two numbers – there are conditions under which accuracy improves while F-measure may drop or be unchanged.

6.2 Experimental Setup

We used a manually constructed data set as gold standard for evaluation. First we randomly sampled 7,000 queries from search engine's daily query logs of different time periods, and had them manually labeled by two annotators independently. Each query is attached to a truth, which is either the query itself for valid queries, or a spelling correction for misspelled ones. From the annotation results that both annotators agreed with each other, we extracted 2,323 query-truth pairs as training set and 991 as test set. Table 1 shows the statistics of the data sets, in which E_q denotes the error rate of query and E_t denotes the error rate of term.

	# queries	# terms	E_q	E_t
Training set	2,323	6,318	15.0%	5.6%
Test set	991	2,589	12.8%	5.2%

Table 1. Statistics of training set and test set

In the following experiments, at most 50 correction candidates were used in the maximum entropy model for each query if there is no special explanation. The web search results were fetched from MSN's search engine. By default, top 100 retrieved items from the web retrieval results were used to perform feature extraction. A set of query log data spanning 9 months are used for collecting statistics required by the baseline.

6.3 Overall Results

Following the method as described in previous sections, we first ran a group of experiments to evaluate the performance of each model we discussed with default settings. The detailed results are shown in Table 2.

Model	Accuracy	Recall	Precision	F
M0	91.8%	60.6%	62.6%	0.616
M1	93.9%	64.6%	77.4%	0.704
M2	94.7%	66.9%	78.0%	0.720

Table 2. Overall Results

From the table we can observe significant performance boosts on all evaluation metrics of M1 and M2 over M0.

We can achieve 25.6% error rate reduction and 23.6% improvement in precision, as well as 6.6% relative improvement in recall, when adding S1 to M1. Paired t-test gives p-value of 0.002, which is significant to 0.01 level.

M2 can bring additional 13.1% error rate reduction and moderate improvement in precision, as well as 3.6% improvement in recall over M1, with paired t-test showing that the improvement is significant to 0.01 level.

6.4 Impact of Candidate number

Theoretically the number of correction candidates in the confusion set determines the accuracy and recall upper bounds for all models concerned in this paper. Performance might be hurt if we use a too small candidate number, which is because the corrections are separated from the confusion sets. These curves shown in Figure 4 and 5, include both theoretical bound (oracle) and actual performance of our described models. From the chart we can see that our models perform best when N_t is around 50, and when $N_t > 15$ the oracle recall and accuracy almost stay unchanged, thus the actual models' performance only benefits a little from larger N_t values. The missing part of recall is largely due to the fact that we are not able to generate truth candidates for some weird query terms rather than insufficient size of confusion set.

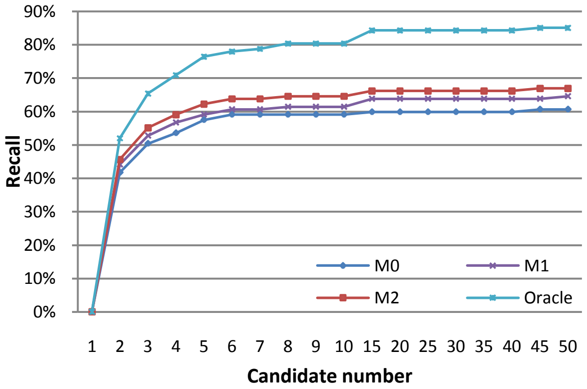


Figure 4. Recall versus candidate number

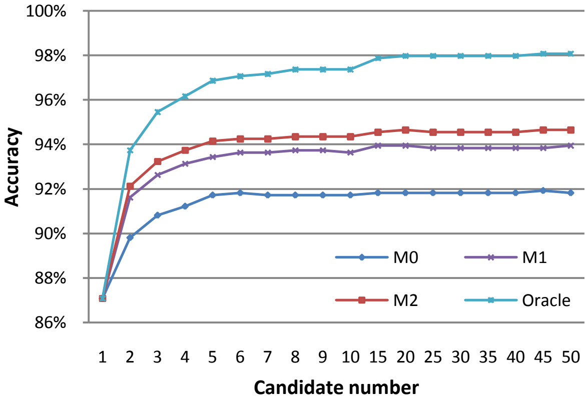


Figure 5. Accuracy versus candidate number

6.5 Discussions

We also studied the performance difference between in-vocabulary (IV) and out-of-vocabulary (OOV) terms when using different spelling correction models. The detailed results are shown in Table 3 and Table 4.

	Accuracy	Precision	Recall	F
M0	88.2%	77.1%	67.3%	0.718
M1	92.4%	88.5%	77.3%	0.825
M2	93.2%	91.6%	79.1%	0.849

Table 3. OOV Term Results

	Accuracy	Precision	Recall	F
M0	98.8%	44.0%	45.8%	0.449
M1	99.0%	62.5%	20.8%	0.313
M2	99.1%	75.0%	37.5%	0.500

Table 4. IV Term Results

The results show that M1 is very powerful to identify and correct OOV spelling errors compared with M0. Actually M1 is able to correct spelling errors such as *guiness*, whose frequency in query log is even higher than its truth spelling *guinness*. Since most spelling errors are OOV terms, this explains why the model M1 can significantly outperform the baseline. But for IV terms things are different. Although the overall accuracy is better, the F-measure of M1 is far lower than M0. M2 performs best for the IV task in terms of both accuracy and F-measure. However, IV spelling errors is so small a portion of the total misspelling (only 17.4% of total spelling errors in our test set) that the room for improvement is very small. This helps to explain why the performance gap between M1 and M0 is much larger than the one between M2 and M1, and shows the tendency that M1 prefer to identify and correct OOV misspellings in comparison to IV ones, which causes F-measure drop from M0 to M1; while by introducing more useful evidence, M2 outperforms better for both OOV and IV terms over M0 and M1.

Another set of statistics we collected from the experiments is the performance data of low-frequency terms when using the models proposed in this paper, since we believe that our approach would help make better classification of low-frequency search terms. As a case study, we identified in the test set all terms whose frequencies in our query logs are less than 800, and for these terms we calculated the error reduction rate of model M1 over the baseline model M0 at each interval of 50. The detailed results are shown in Figure 6. The clear trend can be observed that M1 can achieve larger error rate reduction over baseline for terms with lower frequencies. This is because the performance of baseline model drops for these terms when there are no reliable distributional similarity estimations available due to data sparseness in query logs, while M1 can use web data to alleviate this problem.

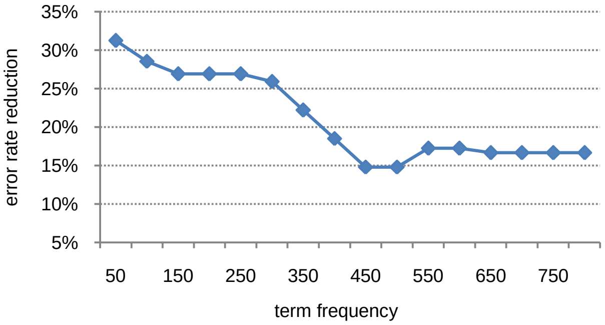


Figure 6. Error rate reduction of M1 over baseline for terms in different frequency ranges

7 Conclusions and Future Work

The task of query spelling correction is very different from conventional spelling checkers, and poses special research challenges. In this paper, we presented a novel method for use of web search results to improve existing query spelling correction models.

We explored two schemes for taking advantage of the information extracted from web search results. Experimental results show that our proposed methods can achieve statistically significant improvements over the baseline model which only relies on evidences of lexicon, spelling similarity and statistics estimated from query logs.

There is still further potential useful information that should be studied in this direction. For example, we can work on page ranking information of returning pages, because trusted or well-known sites with high page rank generally contain few wrong spellings. In addition, the term co-occurrence statistics on the returned snippet text are also worth deep investigation.

References

Ahmad F. and Grzegorz Kondrak G. Learning a spelling error model from search query logs. *Proceedings of EMNLP 2005*, pages 955-962, 2005

Berger A. L., Della Pietra S. A., and Della Pietra V. J. A maximum entropy approach to natural language processing. *Computation Linguistics*, 22(1):39-72, 1996

Brill E. and Moore R. C. An improved error model for noisy channel spelling correction. *Proceedings of 38th annual meeting of the ACL*, pages 286-293, 2000.

Broder, A. A taxonomy of web search. *SIGIR Forum Fall 2002*, Volume 36 Number 2, 2002.

Church K. W. and Gale W. A. Probability scoring for spelling correction. In *Statistics and Computing*, volume 1, pages 93-103, 1991.

Cucerzan S. and Brill E. Spelling correction as an iterative process that exploits the collective knowledge of web users. *Proceedings of EMNLP'04*, pages 293-300, 2004.

Damerau F. A technique for computer detection and correction of spelling errors. *Communication of the ACM* 7(3):659-664, 1964.

Darroch J. N. and Ratcliff D. Generalized iterative scaling for long-linear models. *Annals of Mathematical Statistics*, 43:1470-1480, 1972.

Efthimiadis, N.E., Query Expansion, In *Annual Review of Information Systems and Technology*, Vol. 31, pp. 121-187 , 1996.

Golding A. R. and Roth D. Applying winnow to context-sensitive spelling correction. *Proceedings of ICML 1996*, pages 182-190, 1996.

J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'2001*, pages 111-119, Sept 2001.

J. Xu and W. Croft. Query expansion using local and global document analysis. In *Proceedings of the SIGIR 1996*, pages 4-11, 1996

Kernighan M. D., Church K. W. and Gale W. A. A spelling correction program based on a noisy channel model. *Proceedings of COLING 1990*, pages 205-210, 1990.

Kukich K. Techniques for automatically correcting words in text. *ACM Computing Surveys*. 24(4): 377-439, 1992.

Levenshtein V. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physice – Doklady* 10: 707-710, 1966.

Li M., Zhu M. H., Zhang Y. and Zhou M. Exploring distributional similarity based models for query spelling correction. *Proceedings of COLING-ACL 2006*, pages 1025-1032, 2006

Mangu L. and Eric Brill E. Automatic rule acquisition for spelling correction. *Proceedings of ICML 1997*, pages 734-741, 1997.

Martin Reynaert. Text induced spelling correction. *Proceedings of COLING 2004,*pages 834-840, 2004.

Mayes E., Damerau F. and Mercer R. Context based spelling correction. *Information processing and management,* 27(5): 517-522, 1991.

Philips L. Hanging on the metaphone. *Computer Language Magazine*, 7(12): 39, 1990.

Toutanova K. and Moore R. Pronunciation modeling for improved spelling correction. *Proceedings of the 40th annual meeting of ACL*, pages 144-151, 2002.

Towards Robust Unsupervised Personal Name Disambiguation

Ying Chen
Center for Spoken Language Research
University of Colorado at Boulder
yc@colorado.edu

James Martin
Department of Computer Science
University of Colorado at Boulder
James.Martin@colorado.edu

Abstract

The increasing use of large open-domain document sources is exacerbating the problem of ambiguity in named entities. This paper explores the use of a range of syntactic and semantic features in unsupervised clustering of documents that result from ad hoc queries containing names. From these experiments, we find that the use of robust syntactic and semantic features can significantly improve the state of the art for disambiguation performance for personal names for both Chinese and English.

1 Introduction

An ever-increasing number of question answering, summarization and information extraction systems are coming to rely on heterogeneous sets of documents returned by open-domain search engines from collections over which application developers have no control. A frequent special case of these applications involves queries containing named entities of various sorts and receives as a result a large set of possibly relevant documents upon which further deeper processing is focused. Not surprisingly, many, if not most, of the returned documents will be irrelevant to the goals of the application because of the massive ambiguity associated with the query names of people, places and organizations in large open collections. Without some means of separating documents that contain mentions of distinct entities, most of these applications will produce incorrect results. The work presented here, therefore, addresses the problem of automatically problem of automatically separating sets of news documents generated by queries containing personal names into coherent partitions.

The approach we present here combines unsupervised clustering methods with robust syntactic and semantic processing to automatically cluster returned news documents (and thereby entities) into homogeneous sets. This work follows on the work of Bagga & Baldwin (1998), Mann & Yarowsky (2003), Niu et al. (2004), Li et al. (2004), Pedersen et al. (2005), and Malin (2005). The results described here advance this work through the use of syntactic and semantic features that can be robustly extracted from the kind of arbitrary news texts typically returned from open-domain sources.

The specific contributions reported here fall into two general areas related to robustness. In the first, we explore the use of features extracted from syntactic and semantic processing at a level that is robust to changes in genre and language. In particular, we seek to go beyond the kind of bag of local words features employed in earlier systems (Bagga & Baldwin, 1998; Gooi & Allan, 2004; Pedersen et al., 2005) that did not attempt to exploit deep semantic features that are difficult to extract, and to go beyond the kind of biographical information (Mann & Yarowsky, 2003) that is unlikely to occur with great frequency (such as place of birth, or family relationships) in many of the documents returned by typical search engines. The second contribution involves the application of these techniques to both English and Chinese news collections. As we'll see, the methods are effective with both, but error analyses reveal interesting differences between the two languages.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 190–198, Prague, June 2007. ©2007 Association for Computational Linguistics

The paper is organized as follows. Section 2 addresses related work and compares our work with that of others. Section 3 introduces our new phrase-based features along two dimensions: from syntax to semantics; and from local sentential contexts to document-level contexts. Section 4 first describes our datasets and then analyzes the performances of our system for both English and Chinese. Finally, we draw some conclusions.

2 Previous work

Personal name disambiguation is a difficult problem that has received less attention than those topics that can be addressed via supervised learning systems. Most previous work (Bagga & Baldwin, 1998; Mann & Yarowsky, 2003; Li et al., 2004; Gooi & Allan, 2004; Malin, 2005; Pedersen et al., 2005; Byung-Won On and Dongwon Lee, 2007) employed unsupervised methods because no large annotated corpus is available and because of the variety of the data distributions for different ambiguous personal names.

Since it is common for a single document to contain one or more mentions of the ambiguous personal name of interest, there is a need to define the object to be disambiguated (the ambiguous object). In Bagga & Baldwin (1998), Mann & Yarowsky (2003) and Gooi & Allan (2004), an ambiguous object refers to a single entity with the ambiguous personal name in a given document. The underlying assumption for this definition is "one person per document" (all mentions of the ambiguous personal name in one document refer to the same personal entity in reality). In Niu et al. (2004) and Pedersen et al. (2005), an ambiguous object is defined as a mention of the ambiguous personal name in a corpus.

The first definition of the ambiguous object (document-level object) can include much information derived from that document, so that it can be represented by rich features. The later definition of the ambiguous object (mention-level object) can simplify the detection of the ambiguous object, but because of the limited coverage, it usually can use only local context (the text around the mention of the ambiguous personal name) and might miss some document-level information. The kind of name disambiguation based on mention-level objects really solves "within-document name ambiguity" and "cross-document name ambiguity" simultaneously, and often has a higher performance than the kind based on document-level objects because two mentions of the ambiguous personal name in a document are very likely to refer to the same personal entity. From our news corpus, we also found that mentions of the ambiguous personal name of interest in a news article rarely refer to multiple entities, so our system will focus on the name disambiguation for document-level objects.

In general, there are two types of information usually used in name disambiguation (Malin, 2005): personal information and relational information (explicit and implicit). Personal information gives biographical information about the ambiguous object, and relational information specifies explicit or implicit relations between the ambiguous object and other entities, such as a membership relation between "John Smith" and "Labor Party." Usually, explicit relational information can be extracted from local context, and implicit relational information is far away from the mentions of the ambiguous object. A hard case of name disambiguation often needs implicit relational information that provides a background for the ambiguous object. For example, if two news articles in consideration report an event happening in "Labor Party," this implicit relational information between "John Smith" and "Labor Party" can give a hint for name disambiguation if no personal or explicit relational information is available.

Bagga & Baldwin (1998), Mann & Yarowsky (2003), Gooi & Allan (2004), Niu et al. (2004), and Pedersen et al. (2005) explore features in local context. Bagga & Baldwin (1998), Gooi & Allan (2004), and Pedersen et al. (2005) use local token features; Mann & Yarowsky (2003) extract local biographical information; Niu et al. (2004) use co-occurring Named Entity (NE) phrases and NE relationships in local context. Most of these local contextual features are personal information or explicit relational information.

Li et al. (2004) and Malin (2005) consider named-entity disambiguation as a graph problem, and try to capture information related to the ambiguous object beyond local context, even implicit relational information. Li et al. (2004) use the EM algorithm to learn the global probability distribution among documents, entities, and representative mentions, and Malin (2005) constructs a social network graph to learn a similarity matrix.

In this paper, we also explore both personal and relational information beyond local context. But we achieve it with a different approach: extracting these types of information by means of syntactic and semantic processing. We not only extract local NE phrases as in Niu et al. (2004), but also use our entity co-reference system to extract accurate and representative NE phrases occurring in a document which may have a relation to the ambiguous object. At the same time, syntactic phrase information sometimes can overcome the imperfection of our NE system and therefore makes our disambiguation system more robust.

3 Overall Methodology

Our approach follows a common architecture for named-entity disambiguation: the detection of ambiguous objects, feature extraction and representation, similarity matrix learning, and finally clustering.

In our approach, all documents are preprocessed with a syntactic phrase chunker (Hacioglu, 2004) and the EXERT[1] system (Hacioglu et al. 2005; Chen & Hacioglu, 2006), a named-entity detection and co-reference resolution system that was developed for the ACE[2] project. A syntactic phrase chunker segments a sentence into a sequence of base phrases. A base phrase is a syntactic-level phrase that does not overlap another base phrase. Given a document, the EXERT system first detects all mentions of entities occurring in that document (named-entity detection) and then resolves the different mentions of an entity into one group that uniquely represents the entity (within-document co-reference resolution). The ACE 2005 task can detect seven types of named entities: person, organization, geo-political entity, location, facility, vehicle, and weapon; each type of named entity can occur in a document with any of three distinct formats: name, nominal construction, and pronoun. The F score of the syntactic phrase chunker, which is trained and tested on the Penn TreeBank, is 94.5, and the performances of the EXERT system are 82.9 (ACE value for named-entity detection) and 68.5 (ACE value for within-document co-reference resolution).

3.1 The detection of ambiguous objects

In our approach, we assume that the ambiguous personal name has already been determined by the application. Moreover, we adopt the policy of "one person per document" as in Bagga & Baldwin (1998), and define an ambiguous object as a set of target entities given by the EXERT system. A target entity is an entity that has a mention of the ambiguous personal name. Given the definition of an ambiguous object, we define a local sentence (or local context) as a sentence that contains a mention of any target entity.

3.2 Feature extraction and representation

Since considerable personal and relational information related to the ambiguous object resides in the noun phrases in the document, such as the person's job and the person's location, we attempt to capture this noun phrase information along two dimensions: from syntax to semantics, and from local contexts to document-level contexts.

Base noun phrase feature: To keep this feature focused, we extract only noun phrases occurring in the local sentences and the summarized sentences (the headline + the first sentence of the document) of the document. The local sentences usually include personal or explicit relational information about the ambiguous object, and the summarized sentences of a news document usually give a short summary of the whole news story. With the syntactic phrase chunker, we develop two base noun phrase models: (i) ***Contextual base noun phrases (Contextual bnp)***, the base noun phrases in the local sentences; (ii) ***Summarized base noun phrases (Summarized bnp)***, the base noun phrases in the local sentences and the summarized sentences. A base noun phrase of interest serves as an element in the feature vector.

Named-Entity feature: Given the EXERT system, a direct and simple way to extract some semantic information is to use the named entities detected in the document. Based on their relationship to the ambiguous personal name, the named entities identified in a text can be divided into three categories:

(i) Target entity: an entity that has a mention of the ambiguous personal name. Target entities often include some personal information about the ambiguous object, such as the title, position, and so on.

[1] http://sds.colorado.edu/EXERT

[2] http://projects.ldc.upenn.edu/ace/

Feature Space

***Contextual base noun phrases' feature vector**:* < Hope Mills police Capt. John Smith^{16}, what^{16}, he^{16}, the statements^{16}, no criminal violation^{16}, what^{17}, the individuals^{17}, no direct threat^{17}, Smith^{17}, He and Thomas^{18}, they^{18}, Collins^{18}, his bill^{18}>

***Summarized base noun phrases' feature vector**:* < Hope Mills police Capt. John Smith^{16}, what^{16}, he^{16}, the statements^{16}, no criminal violation^{16}, what^{17}, the individuals^{17}, no direct threat^{17}, Smith^{17}, He and Thomas^{18}, they^{18}, Collins^{18}, his bill^{18}, Collins^{1}, restaurant^{1}, HOPE MILLS^{2}, Commissioner Tonzie Collins^{2}, a town restaurant^{2}, an alleged run-in^{2}, two workers^{2}, Feb. 21^{2}>

Contextual entities' feature vector: **<** Hope Mills police Capt. John Smith^{16}, Jenny Thomas^{4}, Commissioner Tonzie Collins^{2}, He and Thomas^{4}, the individuals^{17}>

Document entities' feature vector: **<** Hope Mills police Capt. John Smith ^{16}, Jenny Thomas^{4}, Commissioner Tonzie Collins^{2}, He and Thomas^{4}, the individuals^{17}, Andy's Cheesesteaks^{4}, HOPE MILLS ^{2}, two workers^{2}, the Village Shopping Center ^{4}, Hope Mills Road ^{4} >

Entity space

Target entity: < Hope Mills police Capt. John Smith^{16}, he^{16}, Smith^{17}, He^{18}>
Local entity: < Thomas^{18}, Jenny Thomas^{4}, manager^{4}>,
< Collins^{18}, his^{18}, Collins^{1}, Commissioner Tonzie Collins ^{2}>, ……
Non-local entity: < restaurant^{1}, a town restaurant^{2}, there^{2}, Andy's Cheesesteaks^{4}>, ……

Text space

(Headline & S1) Collins banned from restaurant
(S2) HOPE MILLS — Commissioner Tonzie Collins has been banned from a town restaurant after an alleged run-in with two workers there Feb. 21. ……
(S4) "In all fairness, that is not a representation of the town," said Jenny Thomas, manager at Andy's Cheesesteaks in the Village Shopping Center on Hope Mills Road. ……
(S16) **Hope Mills police Capt. John Smith** said based on what **he** read in the statements, no criminal violation was committed.
(S17) "Based on what the individuals involved said, there was no direct threat," **Smith** said.
(S18) **He** and Thomas said they don't think Collins intentionally left without paying his bill. ……

Figure 1: A Sample of Feature Extraction

(ii) Local entity: an entity other than a target entity that has a mention occurring in any local sentence. Local entities often include entities that are closely related to the ambiguous object, such as employer, location and co-workers.

(iii) Non-local entity: an entity that is not either the local entity or the target entity. Non-local entities are often implicitly related to the ambiguous object and provide background information for the ambiguous object.

To assess how important these entities are to named-entity disambiguation, we create two kinds of entity models: (i) ***Contextual entities:*** the entities in the feature vector are target entities and local entities; (ii) ***Document entities:*** the entities in the feature vector include all entities in the document including target entities, local entities and non-local entities.

Since a given entity can be represented by many mentions in a document, we choose a single representative mention to represent each entity. The representative mention is selected according to the following ordered preference list: longest NAME mention, longest NOMINAL mention. A representative mention phrase serves as an element in a feature vector.

Although the mentions of contextual entities often overlap with contextual base noun phrases, the representative mention of a contextual entity often goes beyond local sentences, and is usually the first or longest mention of that contextual entity. Compared to contextual base noun phrases, the representative mention of a contextual entity often includes more detail and accurate information about the entity. On the other hand, the contextual base noun phrase feature detects all noun phrases occurring in local sentences that are not limited to the seven types of named entities discovered by the EXERT system. Compared to the contextual entity feature, the contextual base noun phrase feature is more general and can sometimes overcome errors propagated from the named-entity system.

To make this more concrete, the feature vectors for a document containing “John Smith” are highlighted in Figure 1. The superscript number for each phrase refers to the sentence where the phrase is located, and we assume that the syntactic phrase chunker and the EXERT system work perfectly.

3.3 Similarity matrix learning

Given a pair of feature vectors consisting of phrase-based features, we need to choose a similarity scheme to calculate the similarity. Because of the word-space delimiter in English, the feature vector for an English document comprises phrases, whereas that for a Chinese document comprises tokens. There are a number of similarity schemes for learning a similarity matrix from token-based feature vectors, but there are few schemes for phrase-based feature vectors.

Cohen et al. (2003) compared various similarity schemes for the task of matching English entity names and concluded that the hybrid scheme they call SoftTFIDF performs best. SoftTFIDF is a token-based similarity scheme that combines a standard TF-IDF weighting scheme with the Jaro-Winkler distance function. Since Chinese feature vectors are token-based, we can directly use SoftTFIDF to learn the similarity matrix. However, English feature vectors are phrase-based, so we need to run SoftTFIDF iteratively and call it “two-level SoftTFIDF.” First, the standard SoftTFIDF is used to calculate the similarity between phrases in the pair of feature vectors; in the second phase, we reformulate the standard SoftTFIDF to calculate the similarity for the pair of feature vectors.

First, we introduce the standard SoftTFIDF. In a pair of feature vectors *S* and *T*, *S* = ($s_1, \dots, s_n$) and *T* = ($t_1, \dots, t_m$). Here, s_i (i = 1…n) and t_j (j = 1…m) are substrings (tokens). Let *CLOSE*(*θ; S;T*) be the set of substrings $w \in S$ such that there is some $v \in T$ satisfying *dist*(*w; v*) > θ. The Jaro-Winkler distance function (Winkler, 1999) is *dist*(;). For $w \in CLOSE(\theta;\ S;T)$, let *D*(*w; T*) = $\max_{v \in T} dist(w;v)$. Then the standard SoftTFIDF is computed as

$$\mathrm{SoftTFIDF}\,(S,T) =$$

$$\sum\nolimits_{w \in CLOSE(\theta;S;T)} \mathrm{V}(w,\ S) \times \mathrm{V}(w,\ T) \times \mathrm{D}(w,\ T)$$

$$\mathrm{V'}(w,\ S) = \log\,(\mathrm{TF}_{w,S} + 1) \times \log\,(\mathrm{IDF}_{w})$$

$$\mathrm{V}\,(w,\ S) = \frac{\mathrm{V}\ \ (w,\ S)}{\sqrt{\sum\nolimits_{\mathrm{w} \in S} \mathrm{V}\,(w,\ S)^2}}$$

where $\mathrm{TF}_{w,S}$ is the frequency of substrings *w* in *S*, and IDF_w is the inverse of the fraction of documents in the corpus that contain *w*. In computing the similarity for the English phrase-based feature vectors, in the second step of “two-level SoftTFIDF,” the substring *w* is a phrase and *dist* is the standard SoftTFIDF.

So far, we have developed several feature models and learned the corresponding similarity matrices, but clustering usually needs only one unique similarity matrix. Since a feature may have different effects for the disambiguation depending on the ambiguous personal name in consideration, to achieve the best disambiguation ability, each personal name may need its own weighting scheme to combine the given similarity matrices. However, learning that kind of weighting scheme is very difficult, so in this paper, we simply combine the similarity matrices, assigning equal weight to each one.

3.4 Clustering

Although clustering is a well-studied area, a remaining research problem is to determine the optimal parameter setting during clustering, such as the number of clusters or the stop-threshold, a problem that is important for real tasks and that is not at all trivial.

Since the focus of this paper is only on feature development, we simply employ a clustering method that can reflect the quality of the similarity matrix for clustering. Here, we choose agglomerative clustering with a single linkage. Since each personal name may need a different parameter setting, to test the importance of the parameter setting for clustering, we use two kinds of stop-thresholds for agglomerative clustering in our experiments: first, to find the optimal stop-threshold for any ambiguous personal name and for each feature model, we run agglomerative clustering with all possible stop-thresholds, and choose the one that has the best performance as the optimal stop-threshold; second, we use a fixed stop-threshold acquired from development data.

4 Performance

4.1 Data

To capture the real data distribution, we use two sets of naturally occurring data: Bagga's corpus and the Boulder Name corpus, which is a news corpus locally acquired from a web search. Bagga's corpus is a document collection for the English personal name "John Smith" that was used by Bagga & Baldwin (1998). There are 256 articles that match the "/John.*?Smith/" regular expression in 1996 and 1997 editions of the *New York Times*, and 94 distinct "John Smith" personal entities are mentioned. Of these, 83 "John Smiths" are mentioned in only one article (singleton clusters containing only one object), and 11 other "John Smiths" are mentioned several times in the remaining 173 articles (non-singleton clusters containing more than one object). For the task of cross-document co-reference, Bagga & Baldwin (1998) chose 24 articles from 83 singleton clusters, and 173 other articles in 11 non-singleton clusters to create the final test data set – Bagga's corpus.

We collected the Boulder Name corpus by first selecting four highly ambiguous personal names each in English and Chinese. For each personal name, we retrieved the first non-duplicated 100 news articles from Google (Chinese) or Google news (English). There are four data sets for English personal names and four data sets for Chinese personal names: James Jones, John Smith, Michael Johnson, Robert Smith, and Li Gang, Li Hai, Liu Bo, Zhang Yong.

Compared to Bagga's corpus, which is limited to the *New York Times*, the documents in the Boulder Name corpus were collected from different sources, and hence are more heterogeneous and noisy. This variety in the Boulder Name corpus reflects the distribution of the real data and makes named-entity disambiguation harder.

For each ambiguous personal name in both corpora, the gold standard clusters have a long-tailed distribution - a high percentage of singleton clusters plus a few non-singleton clusters. For example, in the 111 documents containing "John Smith" in the Boulder Name corpus, 53 "John Smith" personal entities are mentioned. Of them, 37 "John Smiths" are mentioned only once. The long-tailed distribution brings some trouble to clustering, since in many clustering algorithms a singleton cluster is considered as a noisy point and therefore is ignored.

4.2 Corpus performance

Because of the long tail of the data set, we design a baseline using one cluster per document. To evaluate our disambiguation system, we choose the B-cubed scoring method that was used by Bagga & Baldwin (1998).

In order to compare our work to that of others, we re-implement the model used by Bagga & Baldwin (1998). First, extracting all local sentences produces a summary about the given ambiguous object. Then, the object is represented by the tokens in its summary in the format of a vector, and the tokens in the feature vector are in their morphological root form and are filtered by a stop-word dictionary. Finally, the similarity matrix is learned by the TF-IDF method.

Because both "two-level SoftTFIDF" and agglomerative clustering require a parameter setting, for each language, we reserve two ambiguous personal names from the Boulder Name corpus as development data (John Smith, Michael Johnson, Li Gang, Zhang Yong), and the other data are reserved as test data: Bagga's corpus and the other personal names in the Boulder Name corpus (Robert Smith, James Jones, Li Hai, Liu Bo).

For any ambiguous personal name and for each feature model, we find the optimal stop-threshold for agglomerative clustering, and show the corresponding performances in Table 1, Table 2 and Table 3. However, for the most robust feature model, Bagga + summarized bnp + document entities, we learn the fixed stop-threshold for agglomerative clustering from the development data (0.089 for English data and 0.078 for Chinese data), and show the corresponding performances in Table 4.

4.2.1 Performance on Bagga's corpus

Table 1 shows the performance of each feature model for Bagga's corpus with the optimal stop-threshold. The metric here is the B-cubed F score (precision/recall).

Because of the difference between Bagga's resources and ours (different versions of the named-entity system and different dictionaries of the morphological root and the stop-words), our best

B-cubed F score for Bagga's model is 80.3— 4.3 percent lower than the best performance reported by Bagga & Baldwin (1998): 84.6.

From Table 1, we found that the syntactic features (contextual bnp and summarized bnp) and semantic features (contextual entities and document entities) consistently improve the performances, and all performances outperform the best result reported by Bagga & Baldwin (1998): 84.6

Model	B-cubed performance
Gold standard cluster #	35
Baseline	30.17 (100.00/17.78)
Bagga	80.32 (94.77/69.70)
Bagga + contextual bnp	89.16 (89.18/89.13)
Bagga + summarized bnp	89.59 (92.60/86.78)
Bagga + summarized bnp + contextual entities	89.60 (87.16/92.18)
Bagga + summarized bnp + document entities	92.02 (93.10/90.97)

Table 1: Performances for Bagga's corpus with the optimal stop-threshold

Name Model	John Smith (dev)	Michael Johnson (dev)	Robert Smith (test)	James Jones (test)	Average performance
Gold standard cluster #	53	52	65	24	
Baseline	64.63 (111)	67.97 (101)	78.79 (100)	37.50 (104)	62.22
Bagga	82.63 (75)	89.07 (66)	91.56 (73)	86.42 (24)	87.42
Bagga + contextual bnp	85.18 (62)	89.13 (65)	92.35 (74)	86.45 (22)	88.28
Bagga + summarized bnp	85.97 (66)	91.08 (51)	93.17 (70)	90.11 (33)	90.08
Bagga + summarized bnp + contextual entities	85.44 (70)	94.24 (55)	91.94 (73)	96.66 (24)	92.07
Bagga + summarized bnp + document entities	91.94 (61)	92.55 (51)	93.48 (67)	97.10 (28)	93.77

Table 2: Performances for the English Boulder Name corpus with the optimal stop-threshold

Name Model	Li Gang (dev)	Zhang Yong (dev)	Li Hai (test)	Liu Bo (test)	Average performance
Gold standard cluster #	57	63	57	45	
Baseline	72.61 (100)	76.83 (101)	74.03 (97)	62.07 (100)	71.39
Bagga	96.21 (57)	96.43 (64)	94.51 (64)	91.66 (49)	94.70
Bagga + contextual bnp	97.57 (57)	96.38 (66)	94.53 (64)	93.21 (51)	95.42
Bagga + summarized bnp	98.50 (56)	96.17 (61)	95.38 (62)	93.21 (51)	95.81
Bagga + summarized bnp + contextual entities	99.50 (58)	95.49 (63)	96.75 (58)	91.05 (52)	95.70
Bagga + summarized bnp + document entities	99.50 (56)	94.57 (70)	98.57 (59)	97.02 (48)	97.41

Table 3: Performances for the Chinese Boulder Name corpus with the optimal stop-threshold

English Name	John Smith (dev)	Michael Johnson (dev)	Robert Smith (test)	James Jones (test)	Average performance
Bagga + summarized bnp + document entities	91.31 (*91.94*)	90.57 (*92.55*)	86.71 (*93.48*)	96.64 (*97.10*)	91.31 (*93.77*)
Chinese Name	Li Gang (dev)	Zhang Yong (dev)	Li Hai (test)	Liu Bo (test)	Average performance
Bagga + summarized bnp + document entities	99.06 *(99.50)*	94.56 (*94.56*)	98.25 (*98.57*)	89.18 (*97.02*)	95.26 (*97.41*)

Table 4: Performances for the Boulder Name corpus with the fixed stop-threshold

4.2.2 Performance on the Boulder Name corpus

Table 2 and Table 3 show the performance of each feature model with the optimal stop-threshold for the English and Chinese Boulder Name corpora, respectively. The metric is the B-cubed F score and the number in brackets is the corresponding cluster number. Since the same feature model has different contributions for different ambiguous personal names, we list the average performances for all ambiguous names in the last column in both tables.

Comparing Table 2 and Table 3, we find that Bagga's model has different performances for the English and Chinese corpora. That means that contextual tokens have different contributions in the two languages. There are three apparent causes for this phenomenon. The first concerns the frequency of pronouns in English vs. pro-drop in Chinese. The typical usage of pronouns in English requires an accurate pronoun co-reference resolution that is very important for the local sentence extraction in Bagga's model. In the Boulder Name corpus, we found that ambiguous personal names occur in Chinese much more frequently than in English. For example, the string "Liu Bo" occurs 876 times in the "Liu Bo" data, but the string "John Smith" occurs only 161 times in the "John Smith" data. The repetition of ambiguous personal names in Chinese reduces the burden on pronoun co-reference resolution and hence captures local information more accurately.

The second factor is the fact that tokens in Bagga's model for Chinese are words, but a Chinese word is a unit bigger than an English word, and may contain more knowledge. For example, "the White House" has three words in English, and a word in Chinese. Since Chinese named-entity detection can be considered a sub-problem of Chinese word segmentation, a word in Chinese can catch partial information about named entities.

Finally, compared to Chinese news stories, English news stories are more likely to mention persons marginal to the story, and less likely to give the complete identifying information about them in local context. Those phenomena require more background information or implicit relational information to improve English named-entity disambiguation.

From Table 2 and Table 3, we see that the average performance of all ambiguous personal names is increased (from 87.42 to 93.77 for English and from 94.70 to 97.41 for Chinese) by incorporating more information: contextual bnp (contextual base noun phrases), summarized bnp (summarized base noun phrases), contextual entities, and document entities. This indicates that the phrase-based features, the syntactic and semantic noun phrases, are very useful for disambiguation.

From Table 2 and Table 3, we also see that the phrase-based features can improve the average performance, but not always for all ambiguous personal names. For example, the feature model "Bagga + summarized bnp + contextual entities" hurts the performance for "Robert Smith." As we mentioned above, the Boulder Name corpus is heterogeneous, so each feature does not make the same contribution to the disambiguation for any ambiguous personal name. What we need to do is to find a feature model that is robust for all ambiguous personal names.

In Table 4, we choose the last feature model—Bagga + summarized bnp + document entities—as the final feature model, learn the fixed stop-threshold for clustering from the development data, and show the corresponding performances as B-cubed F scores. The performances in italics are the performances with the optimal stop-threshold. From Table 4, we find that, with the exception of "Robert Smith" and "Liu Bo," the performances for other ambiguous personal names with the fixed threshold are close to the corresponding best performances.

5 Conclusion

This work has explored the problem of personal named-entity disambiguation for news corpora. Our experiments extend token-based information to include noun phrase-based information along two dimensions: from syntax to semantics, and from local sentential contexts to document-level contexts. From these experiments, we find that rich and broad information improves the disambiguation performance considerably for both English and Chinese. In the future, we will continue to explore additional semantic features that can be robustly extracted, including features derived from semantic relations and semantic role labels, and try to extend our work from news articles to web pages that include more noisy information. Finally, we have focused here primarily on feature development and not on clustering. We believe that the skewed long-tailed distribution that characterizes this data requires the use of clustering algorithms tailored to this distribution. In particular, the large number of singleton clusters is an issue that confounds the standard clustering methods we have been employing.

References

A. Bagga and B. Baldwin. 1998. *Entity–based Cross–document Co–referencing Using the Vector Space Model*. In 17th COLING.

Y. Chen and K. Hacioglu. 2006. *Exploration of Coreference Resolution: The ACE Entity Detection and Recognition Task*. In 9th International Conference on TEXT, SPEECH and DIALOGUE.

W. Cohen, P. Ravikumar, S. Fienberg. 2003. *A Comparison of String Metrics for Name-Matching Tasks*. In IJCAI-03 II-Web Workshop.

C. H. Gooi and J. Allan. 2004. *Cross-Document Coreference on a Large Scale Corpus*. In NAACL

K. Hacioglu, B. Douglas and Y. Chen. 2005. *Detection of Entity Mentions Occurring in English and Chinese Text*. Computational Linguistics.

K. Hacioglu. 2004. *A Lightweight Semantic Chunking Model Based On Tagging*. In HLT/NAACL.

X. Li, P. Morie, and D. Roth. 2004. *Robust Reading: Identification and Tracing of Ambiguous Names*. In Proc. of NAACL, pp. 17—24.

B. Malin. 2005. *Unsupervised Name Disambiguation via Social Network Similarity*. SIAM.

G. Mann and D. Yarowsky. 2003. *Unsupervised Personal Name Disambiguation*. In Proc. of CoNLL-2003, Edmonton, Canada.

C. Niu, W. Li, and R. K. Srihari. 2004. *Weakly Supervised Learning for Cross-document Person Name Disambiguation Supported by Information Extraction.* In ACL

B. On and D. Lee. 2007. *Scalable Name Disambiguation using Multi-Level Graph Partition*. SIAM.

T. Pedersen, A. Purandare and A. Kulkarni. 2005. *Name Discrimination by Clustering Similar Contexts*. In Proc. of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics, pages 226-237. Mexico City, Mexico.

T. Pedersen and A. Kulkarni. 2007. *Unsupervised Discrimination of Person Names in Web Contexts*. In Proc. of the Eighth International Conference on Intelligent Text Processing and Computational Linguistics.

W. E. Winkler. 1999. *The state of record linkage and current research problems. Statistics of Income Division*, Internal Revenue Service Publication R99/04.

A. Yates and O. Etzioni. 2007. *Unsupervised Resolution of Objects and Relations on the Web*. In NAACL.

Compressing Trigram Language Models With Golomb Coding

Ken Church
Microsoft
One Microsoft Way
Redmond, WA, USA

Ted Hart
Microsoft
One Microsoft Way
Redmond, WA, USA

Jianfeng Gao
Microsoft
One Microsoft Way
Redmond, WA, USA

{church,tedhar,jfgao}@microsoft.com

Abstract

Trigram language models are compressed using a Golomb coding method inspired by the original Unix spell program. Compression methods trade off space, time and accuracy (loss). The proposed HashTBO method optimizes space at the expense of time and accuracy. Trigram language models are normally considered memory hogs, but with HashTBO, it is possible to squeeze a trigram language model into a few megabytes or less. HashTBO made it possible to ship a trigram contextual speller in Microsoft Office 2007.

1 Introduction

This paper will describe two methods of compressing trigram language models: HashTBO and ZipTBO. ZipTBO is a baseline compression method that is commonly used in many applications such as the Microsoft IME (Input Method Editor) systems that convert Pinyin to Chinese and Kana to Japanese.

Trigram language models have been so successful that they are beginning to be rolled out to applications with millions and millions of users: speech recognition, handwriting recognition, spelling correction, IME, machine translation and more. The EMNLP community should be excited to see their technology having so much influence and visibility with so many people. Walter Mossberg of the Wall Street Journal called out the contextual speller (the blue squiggles) as one of the most notable features in Office 2007:

There are other nice additions. In Word, Outlook and PowerPoint, there is now contextual spell checking, which points to a wrong word, even if the spelling is in the dictionary. For example, if you type "their" instead of "they're," Office catches the mistake. It really works.[1]

The use of contextual language models in spelling correction has been discussed elsewhere: (Church and Gale, 1991), (Mays *et al*, 1991), (Kukich, 1992) and (Golding and Schabes, 1996). This paper will focus on how to deploy such methods to millions and millions of users. Depending on the particular application and requirements, we need to make different tradeoffs among:

1. Space (for compressed language model),
2. Runtime (for *n*-gram lookup), and
3. Accuracy (losses for *n*-gram estimates).

HashTBO optimizes space at the expense of the other two. We recommend HashTBO when space concerns dominate the other concerns; otherwise, use ZipTBO.

There are many applications where space is extremely tight, especially on cell phones. HashTBO was developed for contextual spelling in Microsoft Office 2007, where space was the key challenge. The contextual speller probably would not have shipped without HashTBO compression.

We normally think of trigram language models as memory hogs, but with HashTBO, a few megabytes are more than enough to do interesting things with trigrams. Of course, more memory is always better, but it is surprising how much can be done with so little.

For English, the Office contextual speller started with a predefined vocabulary of 311k word types and a corpus of 6 billion word tokens. (About a

[1] http://online.wsj.com/public/article/SB116786111022966326-T8UUTIl2b10DaW11usf4NasZTYI_20080103.html?mod=tff_main_tff_top

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 199–207, Prague, June 2007.

third of the words in the vocabulary do not appear in the corpus.) The vocabularies for other languages tend to be larger, and the corpora tend to be smaller. Initially, the trigram language model is very large. We prune out small counts (8 or less) to produce a starting point of 51 million trigrams, 14 million bigrams and 311k unigrams (for English). With extreme Stolcke, we cut the 51+14+0.3 million *n*-grams down to a couple million. Using a Golomb code, each *n*-gram consumes about 3 bytes on average.

With so much Stolcke pruning and lossy compression, there will be losses in precision and recall. Our evaluation finds, not surprisingly, that compression matters most when space is tight. Although HashTBO outperforms ZipTBO on the spelling task over a wide range of memory sizes, the difference in recall (at 80% precision) is most noticeable at the low end (under 10MBs), and least noticeable at the high end (over 100 MBs). When there is plenty of memory (100+ MBs), the difference vanishes, as both methods asymptote to the upper bound (the performance of an uncompressed trigram language model with unlimited memory).

2 Preliminaries

Both methods start with a TBO (trigrams with backoff) LM (language model) in the standard ARPA format. The ARPA format is used by many toolkits such as the CMU-Cambridge Statistical Language Modeling Toolkit.[2]

2.1 Katz Backoff

No matter how much data we have, we never have enough. Nothing has zero probability. We will see *n*-grams in the test set that did not appear in the training set. To deal with this reality, Katz (1987) proposed backing off from trigrams to bigrams (and from bigrams to unigrams) when we don't have enough training data.

Backoff doesn't have to do much for trigrams that were observed during training. In that case, the backoff estimate of $P(w_i|w_{i-2}w_{i-1})$ is simply a discounted probability $P_d(w_i|w_{i-2}w_{i-1})$.

The discounted probabilities steal from the rich and give to the poor. They take some probability mass from the rich *n*-grams that have been seen in training and give it to poor unseen *n*-grams that might appear in test. There are many ways to discount probabilities. Katz used Good-Turing smoothing, but other smoothing methods such as Kneser-Ney are more popular today.

Backoff is more interesting for unseen trigrams. In that case, the backoff estimate is:

$$\alpha(w_{i-2}w_{i-1})P_d(w_i|w_{i-1})$$

The backoff alphas (α) are a normalization factor that accounts for the discounted mass. That is,

$$\alpha(w_{i-2}w_{i-1}) = \frac{1-\sum_{w_i:C(w_{i-2}w_{i-1}w_i)} P(w_i|w_{i-2}w_{i-1})}{1-\sum_{w_i:C(w_{i-2}w_{i-1}w_i)} P(w_i|w_{i-1})}$$

where $C(w_{i-2}w_{i-1}w_i) > 0$ simply says that the trigram was seen in training data.

3 Stolcke Pruning

Both ZipTBO and HashTBO start with Stolcke pruning (1998).[3] We will refer to the trigram language model after backoff and pruning as a *pruned TBO LM*.

Stolcke pruning looks for *n*-grams that would receive nearly the same estimates via Katz backoff if they were removed. In a practical system, there will never be enough memory to explicitly materialize all *n*-grams that we encounter during training. In this work, we need to compress a large set of *n*-grams (that appear in a large corpus of 6 billion words) down to a relatively small language model of just a couple of megabytes. We prune as much as necessary to make the model fit into the memory allocation (after subsequent Hash-TBO/ZipTBO compression).

Pruning saves space by removing *n*-grams subject to a loss consideration:

1. Select a threshold θ.
2. Compute the performance loss due to pruning each trigram and bigram individually using the pruning criterion.
3. Remove all trigrams with performance loss less than θ
4. Remove all bigrams with no child nodes (trigram nodes) and with performance loss less than θ
5. Re-compute backoff weights.

[2] http://www.speech.cs.cmu.edu/SLM

[3] http://www.nist.gov/speech/publications/darpa98/html/lm20/lm20.htm

Stolcke pruning uses a loss function based on relative entropy. Formally, let P denote the trigram probabilities assigned by the original unpruned model, and let P' denote the probabilities in the pruned model. Then the relative entropy $D(P||P')$ between the two models is

$$-\sum_{w,h} P(w,h)[\log P'(w|h) - \log P(w,h)]$$

where h is the history. For trigrams, the history is the previous two words. Stolcke showed that this reduces to

$$-P(h)\{P(w|h) \\ [\log P(w|h') + \log \alpha'(h) - \log P(w|h)] \\ +[\log \alpha'(h) - \log \alpha(h)] \sum_{w:C(h,w)>0} P(w|h)\}$$

where $\alpha'(h)$ is the revised backoff weight after pruning and h' is the revised history after dropping the first word. The summation is over all the trigrams that were seen in training: $C(h,w) > 0$.

Stolcke pruning will remove *n*-grams as necessary, minimizing this loss.

3.1 Compression on Top of Pruning

After Stolcke pruning, we apply additional compression (either ZipTBO or HashTBO). ZipTBO uses a fairly straightforward data structure, which introduces relatively few additional losses on top of the pruned TBO model. A few small losses are introduced by quantizing the log likelihoods and the backoff alphas, but those losses probably don't matter much. More serious losses are introduced by restricting the vocabulary size, *V*, to the 64k most-frequent words. It is convenient to use byte aligned pointers. The actual vocabulary of more than 300,000 words for English (and more for other languages) would require 19-bit pointers (or more) without pruning. Byte operations are faster than bit operations. There are other implementations of ZipTBO that make different tradeoffs, and allow for larger *V* without pruning losses.

HashTBO is more heroic. It uses a method inspired by McIlroy (1982) in the original Unix Spell Program, which squeezed a word list of *N*=32,000 words into a PDP-11 address space (64k bytes). That was just 2 bytes per word!

HashTBO uses similar methods to compress a couple million *n*-grams into half a dozen megabytes, or about 3 bytes per *n*-gram on average (including log likelihoods and alphas for backing off). ZipTBO is faster, but takes more space (about 4 bytes per *n*-gram on average, as opposed to 3 bytes per *n*-gram). Given a fixed memory budget, ZipTBO has to make up the difference with more aggressive Stolcke pruning. More pruning leads to larger losses, as we will see, for the spelling application.

Losses will be reported in terms of performance on the spelling task. It would be nice if losses could be reported in terms of cross entropy, but the values output by the compressed language models cannot be interpreted as probabilities due to quantization losses and other compression losses.

4 McIlroy's Spell Program

McIlroy's spell program started with a hash table. Normally, we store the clear text in the hash table, but he didn't have space for that, so he didn't. Hash collisions introduce losses.

McIlroy then sorted the hash codes and stored just the interarrivals of the hash codes instead of the hash codes themselves. If the hash codes, *h*, are distributed by a Poisson process, then the interarrivals, *t*, are exponentially distributed:

$$\Pr(t) = \lambda e^{-\lambda t},$$

where $\lambda = \frac{N}{P}$. Recall that the dictionary contains *N*=32,000 words. *P* is the one free parameter, the range of the hash function. McIlroy hashed words into a large integer mod *P*, where *P* is a large prime that trades off space and accuracy. Increasing *P* consumes more space, but also reduces losses (hash collisions).

McIlroy used a Golomb (1966) code to store the interarrivals. A Golomb code is an optimal Huffman code for an infinite alphabet of symbols with exponential probabilities.

The space requirement (in bits per lexical entry) is close to the entropy of the exponential.

$$H = -\int_{t=0}^{\infty} \Pr(t) \log_2 \Pr(t)\, dt$$

$$\hat{H} = \left\lceil \frac{1}{\log_e 2} + \log_2 \frac{1}{\lambda} \right\rceil$$

The ceiling operator $\lceil\;\rceil$ is introduced because Huffman codes use an integer number of bits to encode each symbol.

We could get rid of the ceiling operation if we replaced the Huffman code with an Arithmetic code, but it is probably not worth the effort.

Lookup time is relatively slow. Technically, lookup time is $O(N)$, because one has to start at the beginning and add up the interarrivals to reconstruct the hash codes. McIlroy actually introduced a small table on the side with hash codes and offsets so one could seek to these offsets and avoid starting at the beginning every time. Even so, our experiments will show that HashTBO is an order of magnitude slower than ZipTBO.

Accuracy is also an issue. Fortunately, we don't have a problem with dropouts. If a word is in the dictionary, we aren't going to misplace it. But two words in the dictionary could hash to the same value. In addition, a word that is not in the dictionary could hash to the same value as a word that is in the dictionary. For McIlroy's application (detecting spelling errors), the only concern is the last possibility. McIlroy did what he could do to mitigate false positive errors by increasing P as much as he could, subject to the memory constraint (the PDP-11 address space of 64k bytes).

We recommend these heroics when space dominates other concerns (time and accuracy).

5 Golomb Coding

Golomb coding takes advantage of the sparseness in the interarrivals between hash codes. Let's start with a simple recipe. Let t be an interarrival. We will decompose t into a pair of a quotient (t_q) and a remainder (t_r). That is, let $t = t_q m + t_r$ where $t_q = \lfloor t/m \rfloor$ and $t_r = t \bmod m$. We choose m to be a power of two near $m \approx \left\lceil \frac{E[t]}{2} \right\rceil = \left\lceil \frac{P}{2N} \right\rceil$, where $E[t]$ is the expected value of the interarrivals, defined below. Store t_q in unary and t_r in binary.

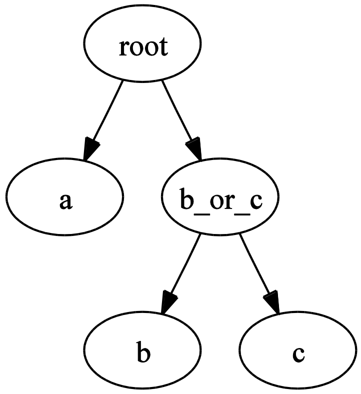
Binary codes are standard, but unary is not. To encode a number z in unary, simply write out a sequence of z-1 zeros followed by a 1. Thus, it takes z bits to encode the number z in unary, as opposed to $\log_2 z$ bits in binary.

This recipe consumes $t_q + \log_2 m$ bits. The first term is for the unary piece and the second term is for the binary piece.

Why does this recipe make sense? As mentioned above, a Golomb code is a Huffman code for an infinite alphabet with exponential probabilities. We illustrate Huffman codes for infinite alphabets by starting with a simple example of a small (very finite) alphabet with just three symbols: $\{a, b, c\}$. Assume that half of the time, we see a, and the rest of the time we see b or c, with equal probabilities:

Symbol	Code	Length	Pr
A	0	1	50%
B	10	2	25%
C	11	2	25%

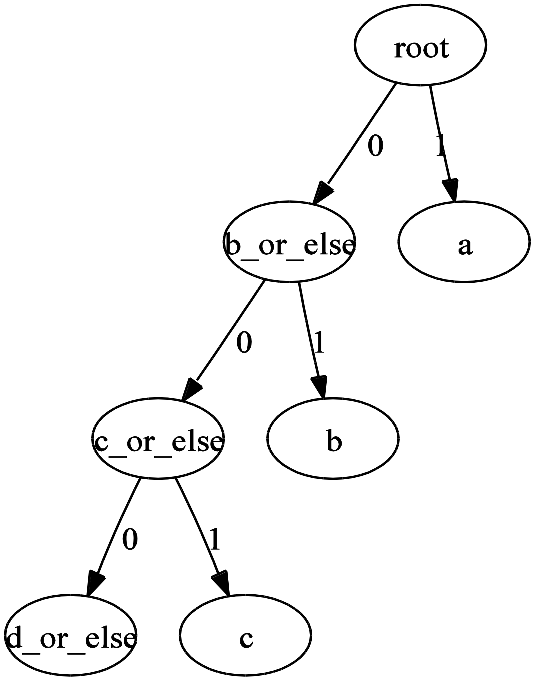
The Huffman code in the table above can be read off the binary tree below. We write out a 0 whenever we take a left branch and a 1 whenever we take a right branch. The Huffman tree is constructed so that the two branches are equally likely (or at least as close as possible to equally likely).



Now, let's consider an infinite alphabet where $\Pr(a) = \frac{1}{2}$, $\Pr(b) = \frac{1}{4}$ and the probability of the *t+1st* symbol is $\Pr(t) = (1-\beta)\beta^t$ where $\beta = \frac{1}{2}$. In this case, we have the following code, which is simply t in unary. That is, we write out $t-1$ zeros followed by a 1.

Symbol	Code	Length	Pr
A	1	1	2^{-1}
B	01	2	2^{-2}
C	001	3	2^{-3}

The Huffman code reduces to unary when the Huffman tree is left branching:



In general, β need not be ½. Without loss of generality, assume $\Pr(t) = (1-\beta)\beta^t$ where $\frac{1}{2} \leq \beta < 1$ and $t \geq 0$. β depends on $E[t]$, the expected value of the interarrivals:

$$E[t] = \frac{P}{N} = \frac{\beta}{1-\beta} \Rightarrow \beta = \frac{E[t]}{1+E[t]}$$

Recall that the recipe above calls for expressing t as $m \cdot t_q + t_r$ where $t_q = \lfloor \frac{t}{m} \rfloor$ and $t_r = t \bmod m$. We encode t_q in unary and t_r in binary. (The binary piece consumes $\log_2 m$ bits, since t_r ranges from 0 to m.)

How do we pick m? For convenience, let m be a power of 2. The unary encoding makes sense as a Huffman code if $\beta^m \approx \frac{1}{2}$.

Thus, a reasonable choice[4] is $m \approx \left\lceil \frac{E[t]}{2} \right\rceil$. If $\beta = \frac{E[t]}{1+E[t]}$, then $\beta^m = \frac{E[t]^m}{(1+E[t])^m} \approx 1 - \frac{m}{E[t]}$. Setting $\beta^m \approx \frac{1}{2}$, means $m \approx \frac{E[t]}{2}$.

6 HashTBO Format

The HashTBO format is basically the same as McIlroy's format, except that McIlroy was storing words and we are storing *n*-grams. One could store all of the n-grams in a single table, though we actually store unigrams in a separate table. An *n*-gram is represented as a key of *n* integers (offsets into the vocabulary) and two values, a log likelihood and, if appropriate, an alpha for backing off. We'll address the keys first.

6.1 HashTBO Keys

Trigrams consist of three integers (offsets into the Vocabulary): $w_1 w_2 w_3$. These three integers are mapped into a single hash between 0 and $P-1$ in the obvious way:

$$hash = (w_3 V^0 + w_2 V^1 + w_1 V^2) \bmod P$$

where V is vocabulary size. Bigrams are hashed the same way, except that the vocabulary is padded with an extra symbol for NA (not applicable). In the bigram case, w_3 is NA.

We then follow a simple recipe for bigrams and trigrams:

1. Stolcke prune appropriately
2. Let *N* be the number of n-grams
3. Choose an appropriate *P* (hash range)
4. Hash the *N* n-grams
5. Sort the hash codes
6. Take the first differences (which are modeled as interarrivals of a Poisson process)
7. Golomb code the first differences

We did not use this method for unigrams, since we assumed (perhaps incorrectly) that we will have explicit likelihoods for most of them and therefore there is little opportunity to take advantage of sparseness.

Most of the recipe can be fully automated with a turnkey process, but two steps require appropriate hand intervention to meet the memory allocation for a particular application:

1. Stolcke prune appropriately, and
2. Choose an appropriate *P*

[4] This discussion follows slide 29 of http://www.stanford.edu/class/ee398a/handouts/lectures/01-EntropyLosslessCoding.pdf. See (Witten *et al*, 1999) and http://en.wikipedia.org/wiki/Golomb_coding, for similar discussion, though with slightly different notation. The primary reference is (Golomb, 1966).

Ideally, we'd like to do as little pruning as possible and we'd like to use as large a *P* as possible, subject to the memory allocation. We don't have a principled argument for how to balance Stolcke pruning losses with hashing losses; this can be arrived at empirically on an application-specific basis. For example, to fix the storage per *n*-gram at around 13 bits:

$$13 = \left\lceil \frac{1}{\log_e 2} + \log_2 \frac{1}{\lambda} \right\rceil$$

If we solve for λ, we obtain $\lambda \approx 1/20{,}0000$. In other words, set *P* to a prime near $20{,}000N$ and then do as much Stolcke pruning as necessary to meet the memory constraint. Then measure your application's accuracy, and adjust accordingly.

6.2 HashTBO Values and Alphas

There are *N* log likelihood values, one for each key. These *N* values are quantized into a small number of distinct bins. They are written out as a sequence of *N* Huffman codes. If there are Katz backoff alphas, then they are also written out as a sequence of *N* Huffman codes. (Unigrams and bigrams have alphas, but trigrams don't.)

6.3 HashTBO Lookup

The lookup process is given an *n*-gram, $w_{i-2} w_{i-1} w_i$, and is asked to estimate a log likelihood, $\log \Pr(w_i | w_{i-2}\, w_{i-1})$. Using the standard backoff model, this depends on the likelihoods for the unigrams, bigrams and trigrams, as well as the alphas.

The lookup routine not only determines if the *n*-gram is in the table, but also determines the offset within that table. Using that offset, we can find the appropriate log likelihood and alpha. Side tables are maintained to speed up random access.

7 ZipTBO Format

ZipTBO is a well-established representation of trigrams. Detailed descriptions can be found in (Clarkson and Rosenfeld 1997; Whittaker and Raj 2001).

ZipTBO consumes 8 bytes per unigram, 5 bytes per bigram and 2.5 bytes per trigram. In practice, this comes to about 4 bytes per n-gram on average.

Note that there are some important interactions between ZipTBO and Stolcke pruning. ZipTBO is relatively efficient for trigrams, compared to bigrams. Unfortunately, aggressive Stolcke pruning generates bigram-heavy models, which don't compress well with ZipTBO.

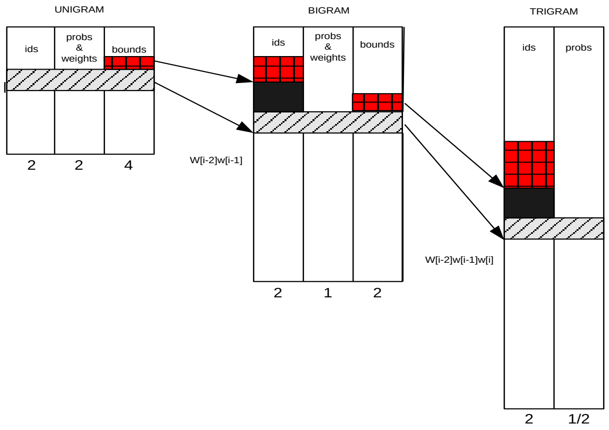


Figure 1. Tree structure of n-grams in ZipTBO format, following Whittaker and Ray (2001)

7.1 ZipTBO Keys

The tree structure of the trigram model is implemented using three arrays. As shown in Figure 1, from left to right, the first array (called *unigram array*) stores unigram nodes, each of which branches out into bigram nodes in the second array (*bigram array*). Each bigram node then branches out into trigram nodes in the third array (*trigram array*).

The length of the unigram array is determined by the vocabulary size (*V*). The lengths of the other two arrays depend on the number of bigrams and the number of trigrams, which depends on how aggressively they were pruned. (We do not prune unigrams.)

We store a 2-byte word id for each unigram, bigram and trigram.

The unigram nodes point to blocks of bigram nodes, and the bigram nodes point to blocks of trigram nodes. There are boundary symbols between blocks (denoted by the pointers in Figure 1). The boundary symbols consume 4 bytes for each unigram and 2 bytes for each bigram.

In each block, nodes are sorted by their word ids. Blocks are consecutive, so the boundary value of an $n-1$-gram node together with the boundary value of its previous $n-1$-gram node specifies, in the n-gram array, the location of the block containing all its child nodes. To locate a particular child node, a binary search of word ids is performed within the block.

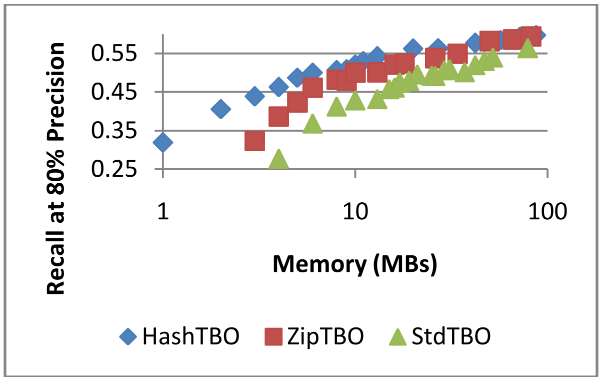


Figure 2. When there is plenty of memory, performance (recall @ 80% precision) asymptotes to the performance of baseline system with no compression (StdTBO). When memory is tight, HashTBO >> ZipTBO >> StdTBO.

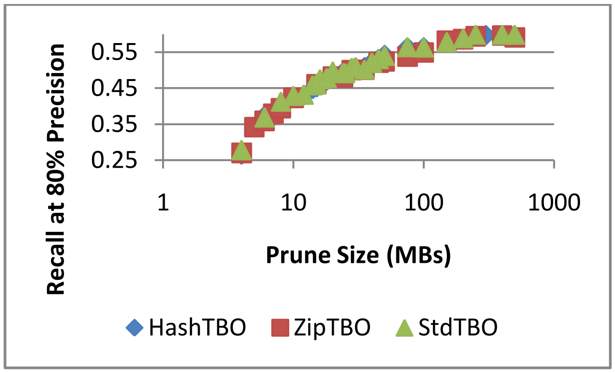


Figure 3. The differences between the methods in Figure 2 vanish if we adjust for prune size.

7.2 ZipTBO Values and Alphas

Like HashTBO, the log likelihood values and backoff alphas are quantized to a small number of quantization levels (256 levels for unigrams and 16 levels for bigrams and trigrams). Unigrams use a full byte for the log likelihoods, plus another full byte for the alphas. Bigrams use a half byte for the log likelihood, plus another half byte for the alphas. Trigrams use a half byte for the log likelihood. (There are no alphas for trigrams.)

7.3 ZipTBO Bottom Line

1. 8 bytes for each unigram:
   a. 2 bytes for a word id +
   b. 4 bytes for two boundary symbols +
   c. 1 byte for a log likelihood +
   d. 1 byte for an alpha
2. 5 bytes for each bigram:
   a. 2 bytes for a word id +
   b. 2 bytes for a boundary symbol +
   c. ½ bytes for a log likelihood +
   d. ½ bytes for an alpha
3. 2.5 bytes for each trigram:
   a. 2 bytes for a word id +
   b. ½ bytes for a log likelihood

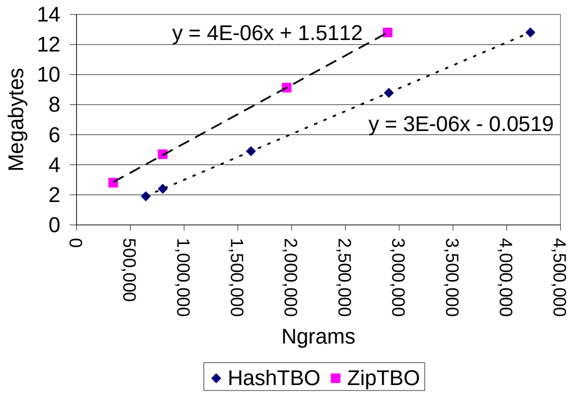


Figure 4. On average, HashTBO consumes about 3 bytes per n-gram, whereas ZipTBO consumes 4.

8 Evaluation

We normally think of trigram language models as memory hogs, but Figure 2 shows that trigrams can be squeezed down to a megabyte in a pinch. Of course, more memory is always better, but it is surprising how much can be done (27% recall at 80% precision) with so little memory.

Given a fixed memory budget, HashTBO outperforms ZipTBO which outperforms StdTBO, a baseline system with no compression. Compression matters more when memory is tight. The gap between methods is more noticeable at the low end (under 10 megabytes) and less noticeable at the high end (over 100 megabytes), where both methods asymptote to the performance of the StdTBO baseline.

All methods start with Stolcke pruning. Figure 3 shows that the losses are largely due to pruning.

All three methods perform about equally well, assuming the same amount of pruning.

The difference is that HashTBO can store more *n*-grams in the same memory and therefore it doesn't have to do as much pruning. Figure 4 shows that HashTBO consumes 3 bytes per *n*-gram whereas ZipTBO consumes 4.

Figure 4 combines unigrams, bigrams and trigrams into a single *n*-gram variable. Figure 5 drills down into this variable, distinguishing bigrams from trigrams. The axes here have been reversed so we can see that HashTBO can store more of both kinds in less space. Note that both HashTBO lines are above both ZipTBO lines.

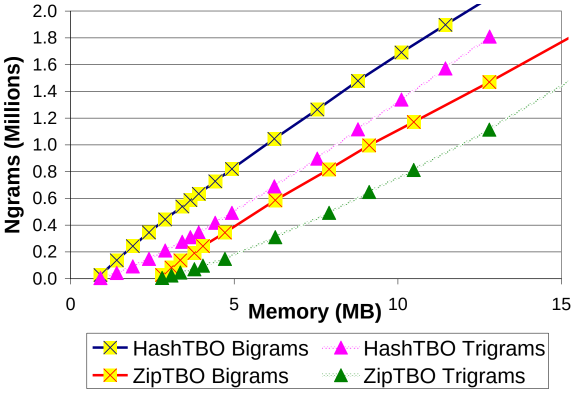


Figure 5. HashTBO stores more bigrams and trigrams than ZipTBO in less space.

In addition, note that both bigram lines are above both trigram lines (triangles). Aggressively pruned models have more bigrams than trigrams!

Linear regression on this data shows that HashTBO is no better than ZipTBO on trigrams (with the particular settings that we used), but there is a big difference on bigrams. The regressions below model *M* (memory in bytes) as a function of *bi* and *tri*, the number of bigrams and trigrams, respectively. (Unigrams are modeled as part of the intercept since all models have the same number of unigrams.)

$$M_{HashTBO} = 0.8 + 3.4bi + 2.6tri$$
$$M_{ZipTBO} = 2.6 + 4.9bi + 2.6tri$$

As a sanity check, it is reassuring that ZipTBO's coefficients of 4.9 and 2.6 are close to the true values of 5 bytes per bigram and 2.5 bytes per trigram, as reported in Section 7.3.

According to the regression, HashTBO is no better than ZipTBO for trigrams. Both models use roughly 2.6 bytes per trigram. When trigram models have relatively few trigrams, the other coefficients matter. HashTBO uses less space for bigrams (3.4 bytes/bigram << 4.9 bytes/bigram) and it has a better intercept (0.8 << 2.6).

We recommend HashTBO if space is so tight that it dominates other concerns. However, if there is plenty of space, or time is an issue, then the tradeoffs work out differently. Figure 6 shows that ZipTBO is an order of magnitude faster than HashTBO. The times are reported in microseconds per n-gram lookup on a dual Xeon PC with a 3.6 ghz clock and plenty of RAM (4GB). These times were averaged over a test set of 4 million lookups. The test process uses a cache. Turning off the cache increases the difference in lookup times.

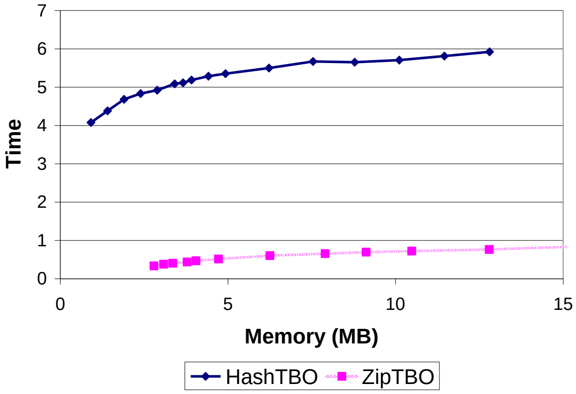


Figure 6. HashTBO is slower than ZipTBO.

9 Conclusion

Trigram language models were compressed using HashTBO, a Golomb coding method inspired by McIlroy's original spell program for Unix. McIlroy used the method to compress a dictionary of 32,000 words into a PDP-11 address space of 64k bytes. That is just 2 bytes per word!

We started with a large corpus of 6 billion words of English. With HashTBO, we could compress the trigram language model into just a couple of megabytes using about 3 bytes per n-gram (compared to 4 bytes per n-gram for the ZipTBO baseline). The proposed HashTBO method is not fast, and it is not accurate (not lossless), but it is hard to beat if space is tight, which was the case for the contextual speller in Microsoft Office 2007.

Acknowledgments

We would like to thank Dong-Hui Zhang for his contributions to ZipTBO.

References

Ashok K. Chandra, Dexter C. Kozen, and Larry J.Stockmeyer. 1981 Alternation. *Journal of the Association for Computing Machinery*, 28(1):114-133.

Church, K., and Gale, W. 1991 Probability Scoring for Spelling Correction, *Statistics and Computing*.

Clarkson, P. and Robinson, T. 2001 Improved language modeling through better language model evaluation measures, *Computer Speech and Language*, 15:39-53, 2001.

Dan Gusfield. 1997 *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK

Gao, J. and Zhang, M., 2002 Improving language model size reduction using better pruning criteria. *ACL 2002*: 176-182.

Gao, J., Goodman, J., and Miao, J. 2001 The use of clustering techniques for language modeling – application to Asian languages. *Computational Linguistics and Chinese Language Processing*, 6:1, pp 27-60.

Golding, A. R. and Schabes, Y. 1996 Combining Trigram-based and feature-based methods for context-sensitive spelling correction, *ACL*, pp. 71-78.

Golomb, S.W. 1966 Run-length encodings *IEEE Transactions on Information Theory*, 12:3, pp. 399-40.

Goodman, J. and Gao, J. 2000 Language model size reduction by pruning and clustering, *ICSLP-2000, International Conference on Spoken Language Processing,* Beijing, October 16-20, 2000.

Mays, E., Damerau, F. J., and Mercer, R. L. 1991 Context based spelling correction. *Inf. Process. Manage.* 27, 5 (Sep. 1991), pp. 517-522.

Katz, Slava, 1987 Estimation of probabilities from sparse data for other language component of a speech recognizer. *IEEE transactions on Acoustics, Speech and Signal Processing,* 35:3, pp. 400-401.

Kukich, Karen, 1992 Techniques for automatically correcting words in text, *Computing Surveys*, 24:4, pp. 377-439.

M. D. McIlroy, 1982 Development of a spelling list, *IEEE Trans. on Communications* 30 pp. 91-99.

Seymore, K., and Rosenfeld, R. 1996 Scalable backoff language models. *Proc. ICSLP*, Vol. 1, pp.232-235.

Stolcke, A. 1998 Entropy-based Pruning of Backoff Language Models. Proc. DARPA News Transcription and Understanding Workshop, 1998, pp. 270--274, Lansdowne, VA.

Whittaker, E. and Ray, B. 2001 Quantization-based language model compression. *Proc. Eurospeech*, pp. 33-36.

Witten, I. H., Moffat, A., and Bell, T. C. 1999 *Managing Gigabytes (2nd Ed.): Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers Inc.

Joint Morphological and Syntactic Disambiguation*

Shay B. Cohen and **Noah A. Smith**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{scohen,nasmith}@cs.cmu.edu

Abstract

In morphologically rich languages, should morphological and syntactic disambiguation be treated sequentially or as a single problem? We describe several efficient, probabilistically-interpretable ways to apply **joint inference** to morphological and syntactic disambiguation using **lattice parsing**. Joint inference is shown to compare favorably to pipeline parsing methods across a variety of component models. State-of-the-art performance on Hebrew Treebank parsing is demonstrated using the new method. The benefits of joint inference are modest with the current component models, but appear to increase as components themselves improve.

1 Introduction

As the field of statistical NLP expands to handle more languages and domains, models appropriate for standard benchmark tasks do not always work well in new situations. Take, for example, parsing the *Wall Street Journal* Penn Treebank, a long-standing task for which highly accurate context-free models stabilized by the year 2000 (Collins, 1999; Charniak, 2000). On this task, the Collins model achieves 90% F_1-accuracy. Extended for new languages by Bikel (2004), it achieves only 75% on Arabic and 72% on Hebrew.[1]

It should come as no surprise that Semitic parsing lags behind English. The Collins model was carefully designed and tuned for *WSJ* English. Many of the features in the model depend on English syntax or Penn Treebank annotation conventions. Inherent in its crafting is the assumption that a million words of training text are available. Finally, for English, it need not handle morphological ambiguity.

Indeed, the figures cited above for Arabic and Hebrew are achieved using *gold-standard* morphological disambiguation and part-of-speech tagging. Given only surface *words*, Arabic performance drops by 1.5 F_1 points. The Hebrew Treebank (unlike Arabic) is built over *morphemes*, a convention we view as sensible, though it complicates parsing.

This paper considers parsing for *morphologically rich* languages, with Hebrew as a test case. Morphology and syntax are two levels of linguistic description that interact. This interaction, we argue, can affect disambiguation, so we explore here the matter of joint disambiguation. This involves the comparison of a **pipeline** (where morphology is inferred first and syntactic parsing follows) with **joint inference**. We present a generalization of the two, and show new ways to do joint inference for this task that does not involve a computational blow-up.

The paper is organized as follows. §2 describes the state of the art in NLP for Hebrew and some phenomena it exhibits that motivate joint inference for morphology and syntax. §3 describes our approach to joint inference using **lattice parsing**, and gives three variants of weighted lattice parsing with their probabilistic interpretations. The different factor models and their stand-alone performance are given in §4. §5 presents experiments on Hebrew parsing and explores the benefits of joint inference.

2 Background

In this section we discuss prior work on statistical morphological and syntactic processing of Hebrew and motivate the joint approach.

2.1 NLP for Modern Hebrew

Wintner (2004) reviews work in Hebrew NLP, emphasizing that the challenges stem from the writing system, rich morphology, unique word formation process of roots and patterns, and relative lack of annotated corpora.

We know of no publicly available statistical parser designed specifically for Hebrew. Sima'an et al.

* The authors acknowledge helpful feedback from the anonymous reviewers, Sharon Goldwater, Rebecca Hwa, Alon Lavie, and Shuly Wintner.

[1] Compared to the Penn Treebank, the Arabic Treebank (Maamouri et al., 2004) has 60% as many word tokens, and the Hebrew Treebank (Sima'an et al., 2001) has 6%.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 208–217, Prague, June 2007.

a. הרועה | באחו | הירוק | הגדול | והמרוחק | שרועה | שם | יפה

b. ה+ רועה ב+ה+ אחו ה+ ירוק ה+ גדול ו+ ה+ מרוחק ש+ רועה שם יפה

c. is-beautiful there shepherds that distant the and big the green the meadow the in shepherd the
ADJ+Masc VB+Masc +Masc

d. the shepherd in the big green distant meadow who shepherds there is beautiful

e. ה+ רועה ב+ה+ אחו ה+ ירוק ה+ גדול ו+ ה+ מרוחק שרועה שם יפה

f. nicely there is-lying distant the and big the green the meadow the in shepherd the
ADV VB+Fem +Fem

g. the shepherdess in the big green distant meadow is lying there nicely

h.

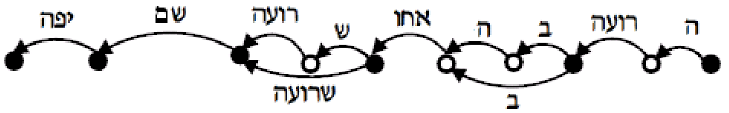


Figure 1: (a.) A sentence in Hebrew (to be read right to left), with (b.) one morphological analysis, (c.) English glosses, and (d.) natural translation; and (e.) a different morphological analysis, (f.) English glosses, and (g.) less natural translation. (h.) shows a morphological "sausage" lattice that encodes the morpheme-sequence analyses $L(\vec{x})$ possible for a shortened sentence (unmodified "meadow"). Shaded states are word boundaries, white states are intra-word morpheme boundaries; in practice we add POS tags to the arcs, to permit disambiguation. According to both native speakers we polled, both interpretations are grammatical—note the long-distance agreement required for grammaticality.

(2001) built a Hebrew Treebank of 88,747 words (4,783 sentences) and parsed it using a probabilistic model. However, they assumed that the input to the parser was already (perfectly) morphologically disambiguated. This assumption is very common in multilingual parsing (see, for example, Cowan et al., 2005, and Buchholz et al., 2006).

Until recently, the NLP literature on morphological processing was dominated by the largely non-probabilistic application of finite-state transducers (Kaplan and Kay, 1981; Koskenniemi, 1983; Beesley and Karttunen, 2003) and the largely unsupervised discovery of morphological patterns in text (Goldsmith, 2001; Wicentowski, 2002); Hebrew morphology receives special attention in Levinger et al. (1995), Daya et al. (2004), and Adler and Elhadad (2006). Lately a few supervised *disambiguation* methods have come about, including hidden Markov models (Hakkani-Tür et al., 2000; Hajič et al., 2001), conditional random fields (Kudo et al., 2004; Smith et al., 2005b), and local support vector machines (Habash and Rambow, 2005). There are also morphological disambiguators designed specifically for Hebrew (Segal, 2000; Bar-Haim et al., 2005).

2.2 Why Joint Inference?

In NLP, the separation of syntax and morphology is understandable when the latter is impoverished, as in English. When both involve high levels of ambiguity, this separation becomes harder to justify, as argued by Tsarfaty (2006). To our knowledge, that is the only study to move toward joint inference of syntax and morphology, presenting joint models and testing approximation of these models with two parsers: one a pipeline (segmentation → tagging → parsing), the other involved joint inference of segmentation and tagging, with the result piped to the parser. The latter was slightly more accurate. Tsarfaty discussed but did not carry out joint inference.

In a morphologically rich language, the different morphemes that make up a word can play a variety of different syntactic roles. A reasonable linguistic analysis might not make such morphemes immediate sisters in the tree. Indeed, the convention of the Hebrew Treebank is to place *morphemes* (rather than words) at the leaves of the parse tree, allowing morphemes of a word to attach to different nonterminal parents.[2]

Generating parse trees over morphemes requires the availability of morphological information when parsing. Because this analysis is not in general reducible to sequence labeling (tagging), the problem is different from POS tagging. Figure 1 gives an

[2] The Arabic Treebank, by contrast, annotates words morphologically but keeps the morphemes together as a single node tagged with a POS sequence. In Bikel's Arabic parser, complex POS tags are projected to a small atomic set; it is unclear how much information is lost.

example from Hebrew that illustrates the interaction between morphology and syntax. In this example, we show two interpretations of the surface text, with the first being a more common natural analysis for the sentence. The first and third-to-last words' analyses depend on each other if the resulting analysis is to be the more natural one: for this analysis the first seven words have to be a noun phrase, while for the less common analysis ("lying there nicely") only the first six words compose a noun phrase, with the last two words composing a verb phrase. Consistency depends on a long-distance dependency that a finite-state morphology model cannot capture, but a model that involves syntactic information can. Disambiguating the syntax aids in disambiguating the morphology, suggesting that a joint model will perform both more accurately.

In sum, joint inference of morphology and syntax is expected to allow decisions of both kinds to influence each other, enforce adherence to constraints at both levels, and to diminish the propagation of errors inherent in pipelines.

3 Joint Inference of Morphology and Syntax

We now formalize the problem and supply the necessary framework for performing joint morphological disambiguation and syntactic parsing.

3.1 Notation and Morphological Sausages

Let $\mathcal{X}$ be the language's word vocabulary and $\mathcal{M}$ be its morpheme inventory. The set of valid analyses for a surface word is defined using a morphological lexicon L, which defines $L(x) \subseteq \mathcal{M}^+$. $L(\vec{x}) \subseteq (\mathcal{M}^+)^+$ (sequence of sequences) is the set of whole-sentence analyses for sentence $\vec{x} = \langle x_1, x_2, ..., x_n \rangle$, produced by concatenating elements of $L(x_i)$ in order. $L(\vec{x})$ can be represented as an acyclic lattice with a "sausage" shape familiar from speech recognition (Mangu et al., 1999) and machine translation (Lavie et al., 2004). Fig. 1h shows a sausage lattice for a sentence in Hebrew. We use $\vec{m}$ to denote an element of $L(\vec{x})$ and $\vec{m}_i$ to denote an element of $L(x_i)$; in general, $\vec{m} = \langle \vec{m}_1, \vec{m}_2, ..., \vec{m}_n \rangle$.

We are interested in a function $f : \mathcal{X}^+ \rightarrow (\mathcal{M}^+)^+ \times \mathcal{T}$, where $\mathcal{T}$ is the set of syntactic trees for the language. f can be viewed as a structured classifier. We use $D_G(\vec{m}) \subseteq \mathcal{T}$ to denote the set of valid trees under a grammar G (here, a PCFG with terminal alphabet $\mathcal{M}$) for morpheme sequence $\vec{m}$. To be precise, $f(\vec{x})$ selects a mutually consistent morphological and syntactic analysis from

$$\text{GEN}(\vec{x}) = \{\langle \vec{m}, \tau \rangle \mid \vec{m} \in L(\vec{x}), \tau \in D_G(\vec{m})\}$$

3.2 Product of Experts

Our mapping $f(\vec{x})$ is based on a joint probability model $p(\tau, \vec{m} \mid \vec{x})$ which combines two probability models $p_G(\tau, \vec{m})$ (a PCFG built on the grammar G) and $p_L(\vec{m} \mid \vec{x})$ (a morphological disambiguation model built on the lexicon L). Factoring the joint model into sub-models simplifies **training**, since we can train each model separately, and **inference** (parsing), as we will see later in this section. Factored estimation has been quite popular in NLP of late (Klein and Manning, 2003b; Smith and Smith, 2004; Smith et al., 2005a, *inter alia*).

The most obvious joint parser uses p_G as a conditional model over trees given morphemes and maximizes the joint likelihood:

$$\begin{aligned} & f_{\text{lik}}(\vec{x}) \\ & = \operatorname*{argmax}_{\langle \vec{m}, \tau \rangle \in \text{GEN}(\vec{x})} p_G(\tau \mid \vec{m}) \cdot p_L(\vec{m} \mid \vec{x}) \qquad (1) \\ & = \operatorname*{argmax}_{\langle \vec{m}, \tau \rangle \in \text{GEN}(\vec{x})} \frac{p_G(\tau, \vec{m})}{\sum_{\tau'} p_G(\tau', \vec{m})} \cdot \frac{p_L(\vec{m}, \vec{x})}{\sum_{\vec{m}'} p_L(\vec{m}', \vec{x})} \end{aligned}$$

This is not straightforward, because it involves summing up the trees for each $\vec{m}$ to compute $p_G(\vec{m})$, which calls for the $O(|\vec{m}|^3)$-Inside algorithm to be called on each $\vec{m}$. Instead, we use the joint, $p_G(\tau, \vec{m})$, which, strictly speaking, makes the model deficient ("leaky"), but permits a dynamic programming solution.

Our models will be parametrized using either unnormalized weights (a log-linear model) or multinomial distributions. Either way, both models define *scores* over parts of analyses, and it may be advantageous to give one model relatively greater strength, especially since we often ignore constant normalizing factors. This is known as a **product of experts** (Hinton, 1999), where a new combined distribution over events is defined by multiplying component distributions together and renormalizing. In the present setting, for some value $\alpha \geq 0$,

$$f_{\text{poe},\alpha}(\vec{x}) = \underset{\langle \vec{m},\tau\rangle \in \text{GEN}(\vec{x})}{\text{argmax}} \frac{p_G(\tau, \vec{m}) \cdot p_L(\vec{m} \mid \vec{x})^\alpha}{Z(\vec{x}, \alpha)} \tag{2}$$

where $Z(\vec{x}, \alpha)$ need not be computed (since it is a constant in $\vec{m}$ and τ). α tunes the relative weight of the morphology model with respect to the parsing model. The higher α is, the more we trust the morphology model over the parser to correctly disambiguate the sentence. We might trust one model more than the other for a variety of reasons: it could be more robustly or discriminatively estimated, or it could be known to come from a more appropriate family.

This formulation also generalizes two more naïve parsing methods. If $\alpha = 0$, the morphology is modeled only through the PCFG and p_L is ignored except as a constraint on which analyses $L(\vec{x})$ are allowed (i.e., on the definition of the set $\text{GEN}(\vec{x})$). At the other extreme, as $\alpha \to +\infty$, p_L becomes more important. Because p_L does not predict trees, p_G still "gets to choose" the syntax tree, but in the limit it must find a tree for $\text{argmax}_{\vec{m} \in L(\vec{x})} p_L(\vec{m} \mid \vec{x})$. This is effectively the morphology-first pipeline.[3]

3.3 Parsing Algorithms

To parse, we apply a dynamic programming algorithm in the $\langle$max, +$\rangle$ semiring to solve the $f_{\text{poe},\alpha}$ problem shown in Eq. 4. If p_L is a unigram-factored model, such that for some single-word morphological model υ we have

$$p_L(\vec{m} \mid \vec{x}) = \prod_{i=1}^n \upsilon(\vec{m}_i \mid x_i) \tag{3}$$

then we can implement morpho-syntactic parsing by *weighting* the sausage lattice. Let the weight of each arc that starts an analysis $\vec{m}_i \in L(x_i)$ be equal to $\log \upsilon(\vec{m}_i \mid x_i)$, and let other arcs have weight 0. In the parsing algorithm, the weight on an arc is summed in when the arc is first used to build a constituent.

In general, we would like to define a joint model that assigns (unnormalized) probabilities to elements of $\text{GEN}(\vec{x})$. If p_G is a PCFG and p_L can be described as a weighted finite-state transducer, then this joint model is their *weighted composition*, which is a weighted CFG; call the composed grammar I and its (unnormalized) distribution p_I. Compared to G, I will have many more nonterminals if p_L has a Markov order greater than 0 (unigram, as above). Because parsing runtime depends heavily on the grammar constant (at best, quadratic in the number of nonterminals), parsing with p_I is not computationally attractive.[4] $f_{\text{poe},\alpha}$ is not, then, a scalable solution when we wish to use a morphology model p_L that can make interdependent decisions about different words in $\vec{x}$ in context. We propose two new, efficient dynamic programming solutions for joint parsing.

In the first, we approximate the distribution $p_L(\vec{M} \mid \vec{x})$ using a unigram-factored model of the form in Eq. 3:

$$p'_L(\vec{m} \mid \vec{x}) = \prod_{i=1}^n \underbrace{p_L(\vec{M}_i = \vec{m}_i \mid \vec{x})}_{\text{posterior, depends on all of } \vec{x}} \tag{7}$$

Similar methods were applied by Matsuzaki et al. (2005) and Petrov and Klein (2007) for parsing under a PCFG with nonterminals with latent annotations. Their approach was variational, approximating the true posterior over coarse parses using a sentence-specific PCFG on the coarse nonterminals, created directly out of the true fine-grained PCFG. In our case, we approximate the full distribution over morphological analyses for the sentence by a simpler, sentence-specific unigram model that assumes each word's analysis is to be chosen independently of the others. Note that our *model* (p_L) does not make such an assumption, only the approximate model p'_L does, and the approximation is per-sentence. The idea resembles a mean-field variational approximation for graphical models. Turning to implementation, we can solve for $p_L(\vec{m}_i \mid \vec{x})$ exactly using the forward-backward algorithm. We will call this method $f_{\text{vari},\alpha}$ (see Eq. 5).

A closely related method, applied by Goodman (1996) is called **minimum-risk** decoding. Goodman called it "maximum expected recall" when applying it to parsing. In the HMM community it

[3] There is a slight difference. If no parse tree exists for the p_L-best morphological analysis, then a less probable $\vec{m}$ may be chosen. So as $\alpha \to +\infty$, we can view $f_{\text{lik},\alpha}$ as finding the best *grammatical* $\vec{m}$ and its best tree—not exactly a pipeline.

[4] In prior work involving factored syntax models—lexicalized (Klein and Manning, 2003b) and bilingual (Smith and Smith, 2004)—$f_{\text{poe},1}$ was applied, and the asymptotic runtime went to $O(n^5)$ and $O(n^7)$.

$$f_{\text{poe},\alpha}(\vec{x}) = \operatorname*{argmax}_{\langle\vec{m},\tau\rangle\in\text{GEN}(\vec{x})} \log p_G(\tau,\vec{m}) + \alpha \log p_L(\vec{m} \mid \vec{x}) \quad (4)$$

$$f_{\text{vari},\alpha}(\vec{x}) = \operatorname*{argmax}_{\langle\vec{m},\tau\rangle\in\text{GEN}(\vec{x})} \log p_G(\tau,\vec{m}) + \alpha \textstyle\sum_{i=1}^{n} \log p_L(\vec{m}_i \mid \vec{x}) \quad (5)$$

$$f_{\text{risk},\alpha}(\vec{x}) = \operatorname*{argmax}_{\langle\vec{m},\tau\rangle\in\text{GEN}(\vec{x})} \log p_G(\tau,\vec{m}) + \alpha \textstyle\sum_{i=1}^{n} p_L(\vec{m}_i \mid \vec{x}) \quad (6)$$

is sometimes called "posterior decoding." Minimum risk decoding is attributable to Goel and Byrne (2000). Applied to a single model, it factors the parsing decision by penalizable errors, and chooses the solution that minimizes the risk (expected number of errors under the model). This factors into a sum of expectations, one per potential mistake. This method is expensive for parsing models (since it requires the Inside algorithm to compute expected recall mistakes), but entirely reasonable for sequence labeling models. The idea is to score each word-analysis $\vec{m}_i$ in the morphological lattice by the expected value (under p_L) that $\vec{m}_i$ is present in the final analysis $\vec{m}$. This is, of course $p_L(\vec{M}_i = \vec{m}_i \mid \vec{x})$, the same quantity computed for $f_{\text{vari},\alpha}$, except the score of a path in the lattice is now a *sum* of posteriors rather than a *product*. Our second approximate joint parser tries to maximize the *probability* of the parse (as before) and at the same time to minimize the *risk* of the morphological analysis. See $f_{\text{risk},\alpha}$ in Eq. 6; the only difference between $f_{\text{risk},\alpha}$ and $f_{\text{vari},\alpha}$ is whether posteriors are added ($f_{\text{risk},\alpha}$) or multiplied ($f_{\text{vari},\alpha}$).

To summarize this section, $f_{\text{vari},\alpha}$ and $f_{\text{risk},\alpha}$ are two approximations to the expensive-in-general $f_{\text{poe},\alpha}$ that boil down to parsing over weighted lattices. The only difference between them is how the lattice is weighted: using $\alpha \log p_L(\vec{m}_i \mid \vec{x})$ for $f_{\text{vari},\alpha}$ or using $\alpha p_L(\vec{m}_i \mid \vec{x})$ for $f_{\text{risk},\alpha}$.[5] In case of a unigram p_L, $f_{\text{poe},\alpha}$ is equivalent to $f_{\text{vari},\alpha}$; otherwise $f_{\text{poe},\alpha}$ is likely to be too expensive.

3.4 Lattice Parsing

To parse the weighted lattices using $f_{\text{vari},\alpha}$ and $f_{\text{risk},\alpha}$ in the previous section, we use *lattice parsing*. Lattice parsing is a straightforward generalization of string parsing that indexes constituents by states in the lattice rather than word interstices. At parsing time, a ⟨max, +⟩ lattice parser finds the best combined parse tree and path through the lattice. Importantly, the data structures that are used in chart parsing need not change in order to accommodate lattices. The generalization over classic Earley or CKY parsing is simple: keep in the parsing chart constituents created over a pair of start state and end state (instead of start position and end position), and (if desired) factor in weights on lattice arcs; see Hall (2005).

4 Factored Models

A fair comparison of joint and pipeline parsing must make some attempt to control for the component models. We describe here two PCFGs we used for $p_G(\tau, \vec{m})$ and two finite-state morphological models we used for $p_L(\vec{m} \mid \vec{x})$. We show how these models perform in stand-alone evaluations. For all experiments, we used the Hebrew Treebank (Sima'an et al., 2001). After removing traces and removing functional information from the nonterminals, we had 3,770 sentences in the training set, 371 sentences in the development set (used primarily to select the value of α) and 370 sentences in the test set.

4.1 Syntax Model

Our first syntax model is an unbinarized PCFG trained using relative frequencies. Preterminal (POS tag → morpheme) rules are smoothed using backoff to a model that predicts the morpheme length and letter sequence. The PCFG is not binarized. This grammar is remarkably good, given the limited effort that went into it. The rules in the training set had **high coverage** with respect to the development set: an oracle experiment in which we maximized the number of recovered gold-standard constituents (on the development set) gave F_1 accuracy of 93.7%. In fact, its accuracy supersedes

[5] Until now, we have talked about weighting *word analyses*, which may cover several arcs, rather than arcs. In practice we apply the weight to the first arc of a word analysis, and weight the remaining arcs of that analysis with 0 (no cost or benefit), giving the desired effect.

more complex, lexicalized, models: given gold-standard morphology, it achieves 81.2% (compared to 72.0% by Bikel's parser, with head rules specified by a native speaker). This is probably attributable to the dataset's size, which makes training with highly-parameterized lexicalized models precarious and prone to overfitting. With first-order **vertical markovization** (i.e., annotating each nonterminal with its parent as in Johnson, 1998), accuracy is also at 81.2%. Tuning the horizontal markovization of the grammar rules (Klein and Manning, 2003a) had a small, adverse effect on this dataset.

Since the PCFG model was relatively successful compared to lexicalized models, and is faster to run, we decided to use a vanilla PCFG, denoted G_{van}, and a parent-annotated version of that PCFG (Johnson, 1998), denoted $G_{\text{v}=2}$.

4.2 Morphology Model

Both of our morphology models use the same morphological lexicon L, which we describe first.

4.2.1 Morphological Lexicon

In this work, a morphological analysis of a word is a sequence of morphemes, possibly with a tag for each morpheme. There are several available analyzers for Hebrew, including Yona and Wintner (2005) and Segal (2000). We use instead an empirically-constructed generative lexicon that has the advantage of matching the Treebank data and conventions. If the Treebank is enriched, this would then directly benefit the lexicon and our models.

Starting with the training data from the Hebrew Treebank, we first create a set of **prefixes** $\mathcal{M}_{\text{p}} \subset \mathcal{M}$; this set includes any morpheme seen in a non-final position within any word. We also create a set of **stems** $\mathcal{M}_{\text{s}} \subset \mathcal{M}$ that includes any morpheme seen in a final position in a word. This effectively captures the morphological analysis convention in the Hebrew Treebank, where a stem is prefixed by a relatively dominant low-entropy sequence of 0–5 prefix morphemes. For example, *MHKLB* ("from the dog") is analyzed as *M+H+KLB* with prefixes *M* ("from") and *H* ("the") and *KLB* ("dog") is the stem. In practice, $|\mathcal{M}_{\text{p}}| = 124$ (including some conventions for numerals) and $|\mathcal{M}_{\text{s}}| = 13{,}588$. The morphological lexicon is then defined as any analysis given $\mathcal{M}_{\text{p}}$ and $\mathcal{M}_{\text{s}}$:

$$\begin{aligned} L(x) &= \{m_1^k \in \mathcal{M}_{\text{p}}^* \times \mathcal{M}_{\text{s}} \mid \text{concat}(m_1^k) = x)\} \\ &\quad \cup \{m_1^k \mid \text{count}(m_1^k, x) \geq 1\} \end{aligned} \tag{9}$$

where m_1^k denotes $\langle m_1, ..., m_k \rangle$ and $\text{count}(m_1^k, x)$ denotes the number of occurrences of x disambiguated as m_1^k in the training set. Note that $L(x)$ also includes any analysis of x observed in the training data. This permits the memorization of any observed analysis that is more involved than simple segmentation (4% of word tokens in the training set; e.g., *LXDR* ("to the room") is analyzed as *L+H+XDR*). This will have an effect on evaluation (see §5.1). On the development data, L has 98.6% coverage.

4.2.2 Unigram Baseline

The baseline morphology model, p_L^{uni}, first defines a joint distribution following Eq. 8. The word model factors out when we conditionalize to form $p_L^{\text{uni}}(\langle m_1, ..., m_k \rangle \mid x)$. The prefix sequence model is multinomial estimated by MLE. The stem model (conditioned on the prefix sequence) is smoothed to permit *any* stem that is a sequence of Hebrew characters. On the development data, p_L^{uni} is 88.8% accurate (by word).

4.2.3 Conditional Random Field

The second morphology model, p_L^{crf}, which is based on the same morphological lexicon L, uses a second-order conditional random field (Lafferty et al., 2001) to disambiguate the full sentence by modeling local contexts (Kudo et al., 2004; Smith et al., 2005b). Space does not permit a full description; the model uses all the features of Smith et al. (2005b) except the "lemma" portion of the model, since the Hebrew Treebank does not provide lemmas. The weights are trained to maximize the probability of the correct path through the morphological lattice, conditioned on the lattice. This is therefore a **discriminative** model that defines $p_L(\vec{m} \mid \vec{x})$ directly, though we ignore the normalization factor in parsing.

Until now we have described p_L as a model of morphemes, but this CRF is trained to predict POS tags as well—we can either use the tags (i.e., label the morphological lattice with tag/morpheme pairs,

$$p_L^{\text{uni}}(\langle m_1, m_2, ..., m_k\rangle, x) = \underbrace{p(x \mid \langle m_1, m_2, ..., m_k\rangle)}_{\text{word}} \cdot \underbrace{p(m_k \mid \langle m_1, ..., m_{k-1}\rangle)}_{\text{stem}} \cdot \underbrace{p(\langle m_1, ..., m_{k-1}\rangle)}_{\text{prefix sequence}} \quad (8)$$

so that the lattice parser finds a parse that is consistent under both models), or sum the tags out and let the parser do the tagging. One subtlety is the tagging of words not seen in the training data; for such words an unsegmented hypothesis with tag UNKNOWN is included in the lattice and may therefore be selected by the CRF. On the development data, p_L^{crf} is 89.8% accurate on morphology, with 74.9% fine-grained POS-tagging F_1-accuracy (see §5.1).

Note on generative and discriminative models. The reader may be skeptical of our choice to combine a generative PCFG with a discrimative CRF. We point out that both are used to define conditional distributions over desired "output" structures given "input" sequences. Notwithstanding the fact that the factors can be *estimated* in very different ways, our combination in an exact or approximate product-of-experts is a reasonable and principled approach.

5 Experiments

In this section we evaluate parsing performance, but an evaluation issue is resolved first.

5.1 Evaluation Measures

The "Parseval" measures (Black et al., 1991) are used to evaluate a parser's phrase-structure trees against a gold standard. They compute precision and recall of constituents, each indexed by a label and two endpoints. As pointed out by Tsarfaty (2006), joint parsing of morphology and syntax renders this indexing inappropriate, since it assumes the yields of the trees are identical—that assumption is violated if there are any errors in the hypothesized $\vec{m}$. Tsarfaty (2006) instead indexed by non-whitespace *character* positions, to deal with segmentation mismatches. In general (and in this work) that is still insufficient, since $L(\vec{x})$ may include $\vec{m}$ that are not simply segmentations of $\vec{x}$ (see §4.2.1).

Roark et al. (2006) propose an evaluation metric for comparing a parse tree over a sentence generated by a speech recognizer to a gold-standard parse. As in our case, the hypothesized tree could have a different yield than the original gold-standard parse tree, because of errors made by the speech recognizer. The metric is based on an alignment between the hypothesized sentence and the gold-standard sentence. We used a similar evaluation metric, which takes into account the information about parallel word boundaries as well, a piece of information that does not appear naturally in speech recognition. Given the correct $\vec{m}^*$ and the hypothesis $\hat{\vec{m}}$, we use dynamic programming to find an optimal many-to-many monotonic alignment between the atomic morphemes in the two sequences. The algorithm penalizes each violation (by a morpheme) of a one-to-one correspondence,[6] and each character edit required to transform one side of a correspondence into the other (without whitespace). Word boundaries are (here) known and included as index positions. In the case where $\hat{\vec{m}} = \vec{m}^*$ (or equal up to whitespace) the method is identical to Parseval (and also to Tsarfaty, 2006). POS tag accuracy is evaluated the same way, for the same reasons; we report F_1-accuracy for tagging and parsing.

5.2 Experimental Comparison

In our experiment we vary four settings:

- Decoding algorithm: $f_{\text{poe},\alpha}$, $f_{\text{risk},\alpha}$, or $f_{\text{vari},\alpha}$ (§3.3).
- Syntax model: G_{van} or $G_{\text{v=2}}$ (§4.1).
- Morphology model: p_L^{uni} or p_L^{crf} (§4.2). In the latter case, we can use the scores over morpheme sequences only (summing out tags before lattice parsing; denoted m.-p_L^{crf}) or the full model over morphemes and tags, denoted t.-p_L^{crf}.[7]
- α, the relative strength given to the morphology model (see §3). We tested values of α in $\{0, +\infty\} \cup \{10^q \mid q \in \{0, 1, ..., 16\}\}$. Recall that $\alpha = 0$ ignores the morphology model probabilities altogether (using an unweighted lattice),

[6] That is, in a correspondence of a morphemes in one string with b in the other, the penalty is $a + b - 2$, since the morpheme on each side is not in violation.

[7] One subtlety is that any arc with the UNKNOWN POS tag can be relabeled—to any other tag—by the syntax model, whose preterminal rules are smoothed. This was crucial for $\alpha = +\infty$ (pipeline) parsing with t.-p_L^{crf} as the morphology model, since the parser does not recognize UNKNOWN as a tag.

			tuned α				pipeline ($\alpha \to +\infty$)			
parser	morph. model	syntax model	seg. acc	fine POS F_1	coarse POS F_1	parse F_1	seg. acc	fine POS F_1	coarse POS F_1	parse F_1
$f_{\text{poe},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	88.0	70.6	75.5	59.5	88.5	71.5	76.1	59.8
		$p_{G_{\text{v}=2}}$	88.0	70.7	75.8	60.4	88.6	70.8	75.7	59.9
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	$*$	$*$	$*$	$*$	90.9	75.6	80.2	63.7
		$p_{G_{\text{v}=2}}$	$*$	$*$	$*$	$*$	90.9	75.3	80.2	64.2
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	$*$	$*$	$*$	$*$	90.9	77.2	†81.5	63.0
		$p_{G_{\text{v}=2}}$	$*$	$*$	$*$	$*$	90.9	77.2	†81.5	64.0
$f_{\text{risk},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	87.9	70.9	75.3	58.9	**88.5**	71.5	76.1	59.8
		$p_{G_{\text{v}=2}}$	87.8	**70.9**	75.6	59.5	**88.6**	70.8	75.6	**59.9**
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	89.8	74.5	78.9	62.5	89.8	74.5	78.9	62.4
		$p_{G_{\text{v}=2}}$	89.8	74.3	79.1	63.0	89.8	74.3	79.1	63.0
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	**90.2**	76.6	80.5	62.4	89.9	76.4	80.4	61.6
		$p_{G_{\text{v}=2}}$	**90.2**	76.6	80.5	**63.1**	89.9	76.4	80.4	62.2
$f_{\text{vari},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	88.0	70.6	75.5	59.5	88.5	71.5	76.1	59.8
		$p_{G_{\text{v}=2}}$	88.0	70.7	75.8	60.4	88.6	70.8	75.7	59.9
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	†**91.1**	**75.6**	**80.4**	**64.0**	90.9	74.8	79.3	62.9
		$p_{G_{\text{v}=2}}$	**90.9**	**75.4**	**80.5**	†**64.4**	90.1	74.6	79.5	63.2
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	†**91.3**	†**77.7**	†81.7	63.0	90.9	77.0	†81.3	62.6
		$p_{G_{\text{v}=2}}$	†**91.3**	†77.6	†81.6	63.6	90.9	77.0	†81.3	63.6

Table 1: Results of experiments on Hebrew (test data, max. length 40). This table shows the performance of **joint parsing** (finite α; left) and a **pipeline** ($\alpha \to +\infty$; right). Joint parsing with a non-unigram morphology model is too expensive (marked $*$). Morphological analysis accuracy (by word), fine-grained (full tags) and coarse-grained (only parts of speech) POS tagging accuracy (F_1), and generalized constituent accuracy (F_1) are reported; α was tuned for each of these separately. **Boldface** denotes that figures were significantly better than their counterparts in the same row, under a binomial sign test ($p < 0.05$). † marks the best overall accuracy and figures that are not significantly worse (binomial sign test, $p < 0.05$).

and as $\alpha \to +\infty$ a morphology-first pipeline is approached.

We measured four outcome values: segmentation accuracy (fraction of word tokens segmented correctly), fine- and coarse-grained tagging accuracy,[8] and parsing accuracy. For tagging and parsing, F_1-measures are given, according to the generalized evaluation measure described in §5.1.

5.3 Results

Tab. 1 compares parsing with tuned α values to the pipeline.

The best results were achieved using $f_{\text{vari},\alpha}$, using the CRF and joint disambiguation. Without the CRF (using p_L^{uni}), the difference between the decoding algorithms is less apparent, suggesting an interaction between the sophistication of the components and the best way to decode with them. These results suggest that $f_{\text{vari},\alpha}$, which permits p_L to "veto" any structure involving a morphological analysis for any word that is *a posteriori* unlikely (note that $\log p_L(\vec{m}_i \mid \vec{x})$ can be an arbitrarily large negative number), is beneficial as a "filter" on parses.[9] $f_{\text{risk},\alpha}$, on the other hand, is only allowed to give "bonuses" of up to α to each morphological analysis that p_L believes in; its influence is therefore weaker. This result is consistent with the findings of Petrov et al. (2007) for another approximate parsing task.

The advantage of the parent-annotated PCFG is also more apparent when the CRF is used for morphology, and when α is tuned. All other things equal, then, p_L^{crf} led to higher accuracy all around. Letting the CRF help predict the POS tags helped tagging accuracy but not parsing accuracy.

While the gains over the pipeline are modest, the segmentation, fine POS, and parsing accuracy scores achieved by joint disambiguation with $f_{\text{vari},\alpha}$ with the CRF are significantly better than any of the pipeline conditions.

Interestingly, if we had not tested with the CRF, we might have reached a very different conclusion about the usefulness of tuning α as opposed to a pipeline. With the unigram morphology model, joint parsing frequently *underperforms* the pipeline, sometimes even signficantly. The explanation, we

[8] Although the Hebrew Treebank is small, the size of its POS tagset is large (four times larger than the Penn Treebank), because the tags encode morphological features (gender, person, and number). These features have either been ignored in prior work or encoded differently. In order for our POS-tagging figures to be reasonably comparable to previous work, we include accuracy for coarse-grained tags (only the core part of speech) tags as well as the detailed Hebrew Treebank tags.

[9] Another way to describe this combination is to call it a product of $|\vec{x}|+1$ experts: one for the morphological analysis of each word, plus the grammar. The morphology experts (softly) veto any analysis that is dubious based on surface criteria, and the grammar (softly) vetoes less-grammatical parses.

parser	morph. model	syntax model	seg. acc	fine POS F_1	coarse POS F_1	parse F_1
$f_{\text{risk},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	90.7	73.4	78.5	64.3
		$p_{G_{\text{v=2}}}$	90.2	73.0	78.5	64.9
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	90.7	75.4	80.0	65.2
		$p_{G_{\text{v=2}}}$	90.8	75.1	80.2	65.4
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	91.2	78.1	82.4	65.7
		$p_{G_{\text{v=2}}}$	91.1	78.0	82.2	66.2
$f_{\text{vari},\alpha}$	p_L^{uni}	$p_{G_{\text{van}}}$	90.6	73.2	78.3	63.5
		$p_{G_{\text{v=2}}}$	90.2	72.8	78.4	64.4
	m.-p_L^{crf}	$p_{G_{\text{van}}}$	92.0	76.6	81.5	66.9
		$p_{G_{\text{v=2}}}$	91.9	76.2	81.6	66.9
	t.-p_L^{crf}	$p_{G_{\text{van}}}$	91.8	79.1	83.2	66.5
		$p_{G_{\text{v=2}}}$	91.7	78.7	83.0	67.4

Table 2: **Oracle** results of experiments on Hebrew (test data, max. length 40). This table shows the performance of morphological segmentation, part-of-speech tagging, coarse part-of-speech tagging and parsing when using an oracle to select the best α for each sentence. The notation and interpretation of the numbers are the same as in Tab. 1.

believe, has to do with the ability of the unigram model to estimate a good *distribution* over analyses. While the unigram model is nearly as good as the CRF at picking the right segmentation for a word, joint parsing demands much more. In case the best segmentation does not lead to a grammatical morpheme sequence (under the syntax model), the morphology model needs to be able to give relative strengths to the alternatives. The unigram model is less able to do this, because it ignores the context of the word, and so the benefit of joint parsing is lost.

Most commonly the tuned value of α is around 10 (not shown, to preserve clarity). Because of ignored normalization constants, this does *not* mean that morphology is “10× more important than syntax,” but it *does* mean that, for a particular p_L and p_G, tuning their relative importance in decoding can improve accuracy. In Tab. 2 we show how performance would improve if the oracle value of α was selected for each test-set *sentence*; this further highlights the potential impact of perfecting the tradeoff between models. Of course, selecting α automatically at test-time, per sentence, is an open problem.

To our knowledge, the parsers we have described represent the state-of-the-art in Modern Hebrew parsing. The closest result is Tsarfaty (2006), which we have not directly replicated. Tsarfaty’s model is essentially a pipeline application of $f_{\text{poe},\infty}$ with a grammar like $p_{G_{\text{van}}}$. Her work focused more on the interplay between the segmentation and POS tagging models and the amount of information passed to the parser. Some key differences preclude direct comparison: we modeled fine-grained tags (though we report both kinds of tagging accurcy), we employed a richer morphological lexicon (permitting analyses that are not just segmentation), and a different training/test split and length filter (we used longer sentences). Nonetheless, our conclusions support the argument in Tsarfaty (2006) for more integrated parsing methods.

We conclude that tuning the relative importance of the two models—rather than pipelining to give one infinitely more importance—can provide an improvement on segmentation, tagging, and parsing accuracy. This suggests that future parsing efforts for languages with rich morphology might continue to assume separately-trained (and separately-improved) morphology and syntax components, which would stand to gain from joint decoding. In our experiments, better morphological disambiguation was crucial to getting any benefit from joint decoding. Our result also suggests that exploring new, fully-integrated models (and training methods for them) may be advantageous.

6 Conclusion

We showed that joint morpho-syntactic parsing can improve the accuracy of both kinds of disambiguation. Several efficient parsing methods were presented, using factored state-of-the-art morphology and syntax models for the language under consideration. We demonstrated state-of-the-art performance on and consistent improvements across many settings for Modern Hebrew, a morphologically-rich language with a relatively small treebank.

References

M. Adler and M. Elhadad. 2006. An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In *Proc. of COLING-ACL*.

R. Bar-Haim, K. Sima’an, and Y. Winter. 2005. Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew. In *Proc. of ACL Workshop on Computational Approaches to Semitic Languages*.

K. R. Beesley and L. Karttunen. 2003. *Finite State Morphology*. CSLI.

D. Bikel. 2004. Multilingual statistical parsing engine. http://www.cis.upenn.edu/~dbikel/software.html#stat-parser.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. of DARPA Workshop on Speech and Natural Language*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, U. Penn.

B. Cowan and M. Collins. 2005. Morphology and reranking for the statistical parsing of Spanish. In *Proc. of HLT-EMNLP*.

E. Daya, D. Roth, and S. Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In *Proc. of EMNLP*.

V. Goel and W. Byrne. 2000. Minimum Bayes risk automatic speech recognition. *Computer Speech and Language*, 14(2):115–135.

J. Goldsmith. 2001. Unsupervised learning of the morphology of natural language. *Comp. Ling.*, 27(2):153–198.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*.

N. Habash and O. Rambow. 2005. Arabic tokenization, part-of-speech tagging, and morphological disambiguation in one fell swoop. In *Proc. of ACL*.

J. Hajič, P. Krbec, P. Květoň, K. Oliva, and V. Petkevič. 2001. Serial combination of rules and statistics: A case study in Czech tagging. In *Proc. of ACL*.

D. Z. Hakkani-Tür, K. Oflazer, and G. Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proc. of COLING*.

K. Hall. 2005. *Best-first Word-lattice Parsing: Techniques for Integrated Syntactic Language Modeling*. Ph.D. thesis, Brown University.

G. E. Hinton. 1999. Products of experts. In *Proc. of ICANN*.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Comp. Ling.*, 24(4):613–632.

R. M. Kaplan and M. Kay. 1981. Phonological rules and finite-state transducers. Presented at LSA.

D. Klein and C. D. Manning. 2003a. Accurate unlexicalized parsing. In *Proc. of ACL*, pages 423–430.

D. Klein and C. D. Manning. 2003b. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS 15*.

K. Koskenniemi. 1983. A general computational model of word-form recognition and production. Technical Report 11, University of Helsinki.

T. Kudo, K. Yamamoto, and Y. Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

A. Lavie, S. Wintner, Y. Eytani, E. Peterson, and K. Probst. 2004. Rapid prototyping of a transfer-based Hebrew-to-English machine translation system. In *Proc. of TMI*.

M. Levinger, U. Ornan, and A. Itai. 1995. Learning morpholexical probabilities from an untagged corpus with an application to Hebrew. *Comp. Ling.*, 21:383–404.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *Proc. of NEMLAR*.

L. Mangu, E. Brill, and A. Stolcke. 1999. Finding consensus among words: Lattice-based word error minimization. In *Proc. of ECSCT*.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of HLT-NAACL*.

B. Roark, M. Harper, E. Charniak, B. Dorr, M. Johnson, J. Kahn, Y. Liu, M. Ostendorf, J. Hale, A. Krasnyanskaya, M. Lease, I. Shafran, M. Snover, R. Stewart, and Lisa Yung. 2006. Sparseval: Evaluation metrics for parsing speech. In *Proc. of LREC*.

E. Segal. 2000. A probabilistic morphological analyzer for Hebrew undotted texts. Master's thesis, Technion.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a treebank of modern Hebrew text. *Journal Traitement Automatique des Langues*. Available at http://mila.cs.technion.ac.il.

D. A. Smith and N. A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. of EMNLP*, pages 49–56.

A. Smith, T. Cohn, and M. Osborne. 2005a. Logarithmic opinion pools for conditional random fields. In *Proc. of ACL*.

N. A. Smith, D. A. Smith, and R. W. Tromble. 2005b. Context-based morphological disambiguation with random fields. In *Proc. of HLT-EMNLP*.

R. Tsarfaty. 2006. Integrated morphological and syntactic disambiguation for Modern Hebrew. In *Proc. of COLING-ACL Student Research Workshop*.

R. Wicentowski. 2002. *Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework*. Ph.D. thesis, Johns Hopkins U.

S. Wintner. 2004. Hebrew computational linguistics: Past and future. *Art. Int. Rev.*, 21(2):113–138.

S. Yona and S. Wintner. 2005. A finite-state morphological grammar of Hebrew. In *Proc. of ACL Workshop on Computational Approaches to Semitic Languages*.

Unsupervised Part-of-Speech Acquisition for Resource-Scarce Languages

Sajib Dasgupta and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{sajib,vince}@hlt.utdallas.edu

Abstract

This paper proposes a new bootstrapping approach to unsupervised part-of-speech induction. In comparison to previous bootstrapping algorithms developed for this problem, our approach aims to improve the quality of the seed clusters by employing seed words that are both distributionally and morphologically reliable. In particular, we present a novel method for combining morphological and distributional information for seed selection. Experimental results demonstrate that our approach works well for English and Bengali, thus providing suggestive evidence that it is applicable to both morphologically impoverished languages and highly inflectional languages.

1 Introduction

The availability of a high-quality lexicon is crucial to the development of fundamental text-processing components such as part-of-speech (POS) taggers and syntactic parsers. While hand-crafted lexicons are readily available for resource-rich languages such as English, the same is not true for *resource-scarce* languages. Unfortunately, manually constructing a lexicon requires a lot of linguistic expertise, and is practically infeasible for highly inflectional and agglutinative languages, which contain a very large number of lexical items. Given the scarcity of annotated data for acquiring the lexicon in a supervised manner, researchers have instead investigated *unsupervised* POS induction techniques for automating the lexicon construction process. In essence, the goal of unsupervised POS induction is to learn the set of possible POS tags for each lexical item from an unannotated corpus.

The most common approach to unsupervised POS induction to date has been motivated by Harris's (1954) distributional hypothesis: words with similar co-occurrence patterns should have similar syntactic behavior. More specifically, unsupervised POS induction algorithms typically operate by (1) representing each *target* word (i.e., a word to be tagged with its POS) as a *context* vector that encodes its left and right context, (2) clustering distributionally similar words, and (3) manually labeling each cluster with a POS tag by inspecting the members of the cluster.

This *distributional* approach works under the assumption that the context vector of each word encodes sufficient information for enabling accurate word clustering. However, many words are *distributionally unreliable*: due to data sparseness, they occur infrequently and hence their context vectors do not capture reliable statistical information. To overcome this problem, Clark (2000) proposes a bootstrapping approach, in which he (1) clusters the most distributionally reliable words, and then (2) incrementally augments each cluster with words that are distributionally similar to those already in the cluster.

The goal of this paper is to propose a new bootstrapping approach to unsupervised POS induction that can operate in a resource-scarce setting. Most notably, our approach aims to improve the quality of the seed clusters by employing seed words that are both *distributionally* and *morphologically* reliable. In particular, we present a novel method for combining morphological and distributional information for seed selection. Furthermore, given our

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 218–227, Prague, June 2007.

emphasis on resource-scarce languages, our approach does not rely on any language resources. In particular, the morphological information that it exploits is provided by an unsupervised morphological analyzer.

It is perhaps not immediately clear why morphological information would play a crucial role in the induction process, especially since the distributional approach has achieved considerable success for English POS induction (see Lamb (1961), Schütze (1995) and Clark (2000)). To understand the role and significance of morphology, it is important to first understand why the distributional approach works well for English. Recall from the above that the distributional approach assumes that the information encoded in the context vector of each word, which typically consists of the 250 most frequent words of a given language, is sufficient for accurately clustering the words. This approach works well for English because the most frequent English words are composed primarily of closed-class words such as "to" and "is", which provide strong clues to the POS of the target word. However, this assumption is not necessarily valid for fairly free word order and highly inflectional languages such as Bengali. The reason is that (1) co-occurrence statistics collected from free word order languages are not as reliable as those from fixed word order languages; and (2) many of the closed-class words that appear in the context vector for English words are realized as inflections in Bengali. The absence of these highly informative words implies that the context vectors may no longer capture sufficient information for accurately clustering Bengali words, and hence the use of morphological information becomes particularly important for unsupervised POS induction for these inflectional languages.

We will focus primarily on labeling *open-class* words with their POS tags. Our decision is motivated by the fact that closed-class words generally comprise a small percentage of the lexical items of a language. In Bengali, the percentage of closed-class words is even smaller than that in English: as mentioned before, many closed-class words in English are realized as suffixes in Bengali.

Although our attempt to incorporate morphological information into the distributional POS induction framework was originally motivated by inflectional languages, experimental results show that our approach works well for both English and Bengali, suggesting its applicability to both morphologically impoverished languages and highly inflectional languages. Owing to the lack of publicly available resources for Bengali, we manually created a 5000-word Bengali lexicon for evaluation purposes. Hence, one contribution of our work lies in the creation of an annotated dataset for Bengali. By making this dataset publicly available[1], we hope to facilitate the comparison of different unsupervised POS induction algorithms and to stimulate interest in Bengali language processing.

The rest of the paper is organized as follows. Section 2 discusses related work on unsupervised POS induction. Section 3 describes our tagsets for English and Bengali. The next three sections describe the three steps of our bootstrapping approach: cluster the words using morphological information (Section 4), remove potentially mislabeled words from each cluster (Section 5), and bootstrap each cluster using a weakly supervised learner (Section 6). Finally, we present evaluation results in Section 7 and conclusions in Section 8.

2 Related Work

Several unsupervised POS induction algorithms have also attempted to incorporate morphological information into the distributional framework, but our work differs from these in two respects.

Computing morphological information. Previous POS induction algorithms have attempted to derive morphological information from dictionaries (Hajič, 2000) and knowledge-based morphological analyzers (Duh and Kirchhoff, 2006). However, these resources are generally not available for resource-scarce languages. Consequently, researchers have attempted to derive morphological information heuristically (e.g., Cucerzan and Yarowsky (2000), Clark (2003), Freitag (2004)). For instance, Cucerzan and Yarowsky (2000) posit a character sequence *x* as a suffix if there exists a sufficient number of distinct words *w* in the vocabulary such that the concatentations *wx* are also in the vocabulary. It is conceivable that such heuristically computed morphological information can be inaccurate, thus rendering the usefulness of a more accurate morphological analyzer. To address this problem, we exploit morphological information provided by an unsupervised word segmentation algorithm.

[1] See http://www.utdallas.edu/~sajib/posDatasets.html.

Tag	Description	Treebank tags
JJ	Adjective	JJ
JJR	Adjective, comparative	JJR
JJS	Adjective, superlative	JJS
NN	Singular noun	NN, NNP
NNS	Plural noun	NNS, NNPS
RB	Adverb	RB
VB	Verb, non-3rd ps. sing. present	VB, VBP
VBD	Verb, past tense or past participle	VBD, VBN
VBG	Verb, gerund/present participle	VBG
VBZ	Verb, 3rd ps. sing. present	VBZ

Table 1: The English tagset

Tag	Description	Examples
JJ	Adjective	vhalo, garam, kharap
NN	Singular noun	kanna, ridoy, shoshon
NN2	2nd order inflectional noun	dhopake, kalamtike
NN6	6th order inflectional noun	gharer, manusher
NN7	7th order inflectional noun	dhakai, barite, graame
NNP	Proper noun	arjun, ahmmad
NNS	Plural noun	manushgulo, pakhider
NNSH	Noun ending with "sh"	barish, jatrish
VB	Finite verb	kheyechi, krlam, krI
VBN	Non-finite verb	kre, giye, jete, kadte

Table 2: The Bengali tagset

Using morphological information. Perhaps due to the overly simplistic methods employed to compute morphological information, morphology has only been used as what Biemann (2006) called *add-on's* in existing POS induction algorithms, which remain primarily distributional in nature. In contrast, our approach more tightly integrates morphology into the distributional framework. As we will see, we train SVM classifiers using both morphological and distributional features to select seed words for our bootstrapping algorithm, effectively letting SVM combine these two sources of information and perform automatic feature weighting. Another appealing feature of our approach is that when labeling each unlabeled word with its POS tag, an SVM classifier also returns a numeric value that indicates how confident the word is labeled. This opens up the possibility of having a human improve our automatically constructed lexicon by manually checking those entries that are tagged with low confidence by an SVM classifier.

Recently, there have been attempts to perform (mostly) unsupervised POS tagging without relying on a POS lexicon. Haghighi and Klein's (2006) *prototype-driven* approach requires just a few prototype examples for each POS tag, exploiting these labeled words to constrain the labels of their distributionally similar words when training a generative log-linear model for POS tagging. Smith and Eisner (2005) train a log-linear model for POS tagging in an unsupervised manner using *contrastive estimation*, which seeks to move probability mass to a positive example *e* from its *neighbors* (i.e., negative examples created by perturbing *e*).

3 The English and Bengali Tagsets

Given our focus on automatically labeling open class words, our English and Bengali tagsets are designed to essentially cover all of the open-class words. Our English tagset, which is composed of ten tags, is shown in Table 1. As we can see, a tag in our tagset can be mapped to more than one Penn Treebank tags. For instance, we use the tag "NN" for both singular and plural common nouns. Our decision of which Penn Treebank tags to group together is based on that of Schütze (1995).

Our Bengali tagset, which also consists of ten tags, is adapted from the one proposed by Saha et al. (2004) (see Table 2). It is worth noting that unlike English, we assign different tags to Bengali proper nouns and common nouns. The reason is that for English, it is not particularly crucial to distinguish the two types of nouns during POS induction, since they can be distinguished fairly easily using heuristics such as initial capitalization. For Bengali, such simple heuristics do not exist, as the Bengali alphabet does not have any upper and lower case letters. Hence, it is important to distinguish Bengali proper nouns and common nouns during POS induction.

4 Clustering the Morphologically Similar Words

As mentioned before, our approach aims to more tightly integrate morphological information into the distributional POS induction framework. In fact, our POS induction algorithm begins by clustering the *morphologically similar* words (i.e., words that combine with the same set of suffixes). The motivation for clustering morphologically similar words can be attributed to our hypothesis that words having similar POS should combine with a similar set of suffixes. For instance, verbs in English combine with suffixes like "ing", "ed" and "s", whereas adjectives combine with suffixes like "er" and "est". Note, however, that the suffix "s" can attach to both verbs and nouns in English, and so it is not likely to be a useful feature for identifying the POS of a word. The question, then, is how to determine which suffixes are useful for the POS identification task in an *unsupervised* setting where we do not have any prior knowledge of language-specific grammatical constraints. This section proposes a method for identifying the "useful" suffixes and employing them to cluster the morphologically similar words. As we will see, our clustering algorithm not only produces soft clusters, but it also automatically determines the number of clusters for a particular language.

Before we describe how to identify the useful suffixes, we need to (1) induce all of the suffixes and (2) morphologically segment the words in our vocabulary.[2] However, neither of these tasks is simple for a truly resource-scarce language for which we do not have a dictionary or a knowledge-based morphological analyzer. As mentioned in the introduction, our proposed solution to both tasks is to use an unsupervised morphological analyzer that can be built just from an unannotated corpus. In particular, we have implemented an unsupervised morphological analyzer that outperforms Goldsmith's (2001) Linguistica and Creutz and Lagus's (2005) Morfessor for our English and Bengali datasets and compares favorably to the best-performing morphological parsers in MorphoChallenge 2005[3] (see Dasgupta and Ng (2007)).

Given the segmentation of each word and the most frequent 30 suffixes[4] provided by our morphological analyzer, our clustering algorithm operates by (1) clustering the similar suffixes and then (2) assigning words to each cluster based on the suffixes a word combines with. To cluster similar suffixes, we need to define the similarity between two suffixes. Informally, we say that two suffixes x and y are similar if a word that combines with x also combines with y and vice versa. In practice, we will rarely posit two suffixes as similar under this definition unless we assume access to a complete vocabulary – an assumption that is especially unrealistic for resource-scarce languages. As a result, we relax this definition and consider two suffixes x and y similar if $P(x \mid y) > t$ and $P(y \mid x) > t$, where $P(x \mid y)$ is the probability of a word combining with suffix x given that it combines with suffix y, and t is a threshold that we set to 0.4 in all of our experiments. Note that both probabilities can be estimated from an unannotated corpus.[5] Given this definition of similarity, we can cluster the similar suffixes using the following steps:

Creating the initial clusters. First, we create a *suffix graph*, in which we have (1) one node for each of the 30 suffixes, and (2) a directed edge from suffix x to suffix y if $P(y \mid x) > 0.4$. We then identify the strongly connected components of this graph using depth-first search. These strongly connected components define our initial partitioning of the 30 suffixes. We denote the suffixes assigned to a cluster the *primary keys* of the cluster.

Improving the initial clusters. Recall that we ultimately want to cluster the words by assigning each word w to the cluster in which w combines with all of its primary keys. Given this goal, it is conceivable that singleton clusters are not desirable. For instance, a cluster that has "s" as its only primary key is not useful, because although a lot of words combine with "s", they do not necessarily have the same POS. As a result, we improve each initial cluster by adding more suffixes to the cluster, in hopes of improving the resulting clustering of the words by placing additional constraints on each cluster. More specifically, we add a suffix y to a cluster c if, for each primary key x of c, $P(y \mid x) > 0.4$. If this condition is satisfied, then y becomes a *secondary key* of c. For each initial cluster c', we perform this check using each of the suffixes x' not in c' to see if x' can be added to c'. If, after this expansion step, we still have a cluster c* defined by a single primary key x that also serves as a secondary key in other clusters, then x is *probably ambiguous* (i.e., x can probably attach to words belonging to different POSs); and consequently, we remove c*. We denote the resulting set of clusters by C.

Populating the clusters with words. Next, for each word w in our vocabulary, we check whether w can be assigned to any of the clusters in C. Specifically, we assign w to a cluster c if w can combine with each of its primary keys and at least half of its secondary keys.

Labeling and merging the clusters. After populating each cluster with words, we manually label

[2] A vocabulary is simply a set of (distinct) words extracted from an unannotated corpus. We extracted our English and Bengali vocabulary from WSJ and Prothom Alo, respectively.

[3] http://www.cis.hut.fi/morphochallenge2005/

[4] We found that 30 suffixes are sufficient to cluster the words.

[5] For instance, we compute $P(x \mid y)$ as the ratio of the number of distinct words that combines with both x and y to the number of distinct words that combine with y only.

each of them with a POS tag from the tagset. We found that all of the clusters are labeled as NN, VB, or JJ. The reason is that the clustered words are mostly root words. We then merge all the clusters labeled with the same POS tag, yielding only three "big" clusters. Note that these "big" clusters are *soft* clusters, since a word can belong to more than one of them. For instance, "cool" can combine with "s" or "ing" to form a VB, and it can also combine with "er" or "est" to form a JJ.

Generating sub-clusters. Recall that each "big" cluster contains a set of suffixes and also a set of words that combines with those suffixes. Now, for each "big" cluster c, we create one sub-cluster c_x for each suffix x that appears in c. Then, for each word w in c, we use our unsupervised morphological analyzer to generate $w+x$ and add the surface form to the corresponding sub-cluster.

Labeling the sub-clusters. Finally, we manually label each sub-cluster with a POS tag from our tagset. For example, all the words ending in "ing" will be labeled as VBG. As before, we merge two clusters if they are labeled with the same POS tag. The resulting clusters are our morphologically formed clusters.

5 Purifying the Seed Set

The clusters formed thus far cannot be expected to be perfectly accurate, since (1) our unsupervised morphological analyzer is not perfect, and (2) morphology alone is not always sufficient for determining the POS of a word. In fact, we found that many adjectives are mislabeled as nouns for both languages. For instance, "historic" is labeled as a noun, since it combines with suffixes like "al" and "ally" that "accident" combines with. In addition, many words are labeled with the POS that does not correspond to their most common word sense. For instance, while words like "chair", "crowd" and "cycle" are more commonly used as nouns than verbs, they are labeled as verbs by our clustering algorithm. The reason is that suffixes that typically attach to verbs (e.g., "s", "ed", "ing") also attach to these words. Such labelings, though not incorrect, are undesirable, considering the fact that these words are to be used as seeds to bootstrap our morphologically formed clusters in a distributional manner. For instance, since "chair" and "crowd" are distributionally similar to nouns, their presence in the verb clusters can potentially contaminate the clusters with nouns during the bootstrapping process. Hence, for the purpose of effective bootstrapping, we also consider these words "mislabeled".

To identify the words that are potentially mislabeled, we rely on the following assumption: words that are morphologically similar should also be distributionally similar and vice versa. Based on this assumption, we propose a *purification* method that posits a word w as potentially mislabeled (and therefore should be removed or relabeled) if the POS of w as predicted using distributional information differs from that as determined by morphology.

The question, then, is how to predict the POS tag of a word using distributional information? Our idea is to use "supervised" learning, where we train and test on the seed set. Conceptually, we (1) train a *multi-class* classifier on the morphologically labeled words, each of which is represented by its context vector, and (2) apply the classifier to relabel the *same* set of words. If the new label of a word w differs from its original label, then morphology and context disagree upon the POS of w; and as mentioned above, our method then determines that the word is potentially misclassified. Note, however, that (1) the training instances are not perfectly labeled and (2) it does not make sense to train a classifier on data that is seriously mislabeled. Hence, we make the assumption that a large percentage (> 70%) of the training instances is correctly labeled[6], and that our method would work with a training set labeled at this level of accuracy. In addition, since we are training a classifier based on distributional features, we train and test on only *distributionally reliable* words, which we define to be words that appear at least five times in our corpus. Distributionally unreliable words will all be removed from the morphologically formed clusters, since we cannot predict their POS using distributional information.

In our implementation of this method, rather than train a multi-class classifier, we train a set of binary classifiers using SVM^{light} (Joachims, 1999) together with the distributional features for determining the POS tag of a given word.[7] More specifically, we train one classifier for each pair of

[6] An inspection of the morphologically formed clusters reveals that this assumption is satisfied for both languages.

[7] In this and all subsequent uses of SVM^{light}, we set all the training parameters to their default values.

POS tags. For instance, since we have ten POS tags for English, we will train 45 binary classifiers.[8] To determine the POS tag of a given English word w, we will use these 45 pairwise classifiers to independently assign a label to w. For instance, the NN-JJ classifier will assign either NN or JJ to w. We then count how many times w is tagged with each of the ten POS tags. If there is a POS tag t whose count is nine, it means that all the nine classifiers associated with t have classified w as t, and so our method will label w as t. Otherwise, we remove w from our seed set, since we cannot confidently label it using our classifier ensemble.

To create the training set for the NN-JJ classifier, for instance, we can possibly use all of the words labeled with NN and JJ as positive and negative instances, respectively. However, to ensure that we do not have a skewed class distribution, we use the same number of instances from each class to train the classifier. More formally, let I_{NN} be the set of instances labeled with NN, and I_{JJ} be the set of instances labeled with JJ. Without loss of generality, assume that $|I_{\mathrm{NN}}| < |I_{\mathrm{JJ}}|$, where $|X|$ denotes the size of the set X. To avoid class skewness, we have to sample from I_{JJ}, since it is the larger set. Our sampling method is motivated by bagging (Breiman, 1996). More specifically, we create 10 training sets from I_{JJ}, each of which has size $|I_{\mathrm{NN}}|$ and is formed by sampling with replacement from I_{JJ}. We then combine each of these 10 training sets separately with I_{NN}, and train 10 SVM classifiers from the 10 resulting training sets. Given a test instance i, we first apply the 10 classifiers independently to i and obtain the signed confidence values[9] of the predictions provided by the classifiers. We then take the average of the 10 confidence values, assigning i the positive class if the average is at least 0, and negative otherwise.

As mentioned above, we use distributional features to represent an instance created from a word w. The distributional features are created based on Schütze's (1995) method. Specifically, the left context and the right context of w are each encoded using the most frequent 500 words from the vocabulary. A feature in the left (right) context has the value 1 if the corresponding word appears to the left (right) of w in our corpus, and 0 otherwise. However, we found that using distributional features alone would erroneously classify words like "car" and "cars" as having the same POS because the two words are distributionally similar. In general, it is difficult to distinguish words in NN from those in NNS by distributional means. The same problem occurs for words in VB and VBD. To address this problem, we augment the feature set with suffixal features. Specifically, we create one binary feature for each of the 30 most frequent suffixes that we employed in the previous section. The feature corresponding to suffix x has the value 1 if x is the suffix of w. Moreover, we create an additional suffixal feature whose value is 1 if none of the 30 most frequent suffixes is the suffix of w.

6 Augmenting the Seed Set

After purification, we have a set of clusters filled with distributionally and morphologically reliable seed words that receive the same POS tag when predicted independently by morphological features and distributional features. Our goal in this section is to augment this seed set. Since we have a small seed set (5K words for English and 8K words for Bengali) and a large number of unlabeled words, we believe that it is most natural to apply a weakly supervised learning algorithm to bootstrap the clusters. Specifically, we employ a version of self-training together with SVM as the underlying learning algorithm.[10] Below we first present the high-level idea of our self-training algorithm and then discuss the implementation details.

Conceptually, our self-training algorithm works as follows. We first train a multi-class SVM classifier on the seed set for determining the POS tag of a word using the morphological and distributional features described in the previous section, and then apply it to label the unlabeled (i.e., unclustered) words. Words that are labeled with a confidence value that exceeds the current threshold (which is initially set to 1 and -1 for positively and negatively labeled instances, respectively) will be

[8] We could have trained just one 10-class classifier, but the fairly large number of classes leads us to speculate that this multi-class classifier will not achieve a high accuracy.

[9] Here, a large positive number indicates that the classifier confidently labels the instance as NN, and a large negative number represents confident prediction for JJ.

[10] As a related note, Clark's (2001) bootstrapping algorithm uses KL-divergence to measure the distributional similarity between an unlabeled word and a labeled word, adding to a cluster the words that are most similar to its current member. For us, SVM is a more appealing option because it automatically combines the morphological and distributional features.

added to the seed set. In the next iteration, we retrain the classifier on the augmented labeled data, apply it to the unlabeled data, and add to the labeled data those instances whose predicted confidence is above the current threshold. If none of the instances has a predicted confidence above the current threshold, we reduce the threshold by 0.1. (For instance, if the original thresholds are 1 and -1, they will be changed to 0.9 and -0.9.) We then repeat the above procedure until the thresholds reach 0.5 and -0.5. [11] Finally, we apply the resulting bootstrapped classifier to label all of the unlabeled words that have a corpus frequency of at least five, using a threshold of 0.

In our implementation of the self-training algorithm, rather than train a multi-class classifier in each bootstrapping iteration, we train pairwise classifiers (recall that for English, 45 classifiers are formed from 10 POS tags) using the morphological and distributional features described in the previous section. Again, since we employ distributional features, we apply the 45 pairwise classifiers only to the distributionally reliable words (i.e., words with corpus frequency at least 5). To classify an unlabeled word *w*, we apply the 45 pairwise classifiers to independently assign a label to *w*.[12] We then count how many times *w* is tagged with each of the ten POS tags. If there is a POS tag whose count is nine and all of these nine votes are associated with confidence that exceeds the current threshold, then we add *w* to the labeled data together with its assigned tag.

7 Evaluation

7.1 Experimental Setup

Corpora. Recall that our bootstrapping algorithm assumes as input an unannotated corpus from which we (1) extract our *vocabulary* (i.e., the set of words to be labeled) and (2) collect the statistics needed in morphological and distributional clustering. We use as our English corpus the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus et al., 1993). Our Bengali corpus is composed of five years of articles taken from the Bengali newspaper *Prothom Alo*.

Vocabulary creation. To extract our English vocabulary, we pre-processed each document in the WSJ corpus by first tokenizing them and then removing the most frequent 500 words (as they are mostly closed class words), capitalized words, punctuations, numbers, and unwanted character sequences (e.g., "***"). The resulting English vocabulary consists of approximately 35K words. We applied similar pre-processing steps to the Prothom Alo articles to generate our Bengali vocabulary, which consists of 80K words.

Test set preparation. Our English test set is composed of the 25K words in the vocabulary that appear at least five times in the WSJ corpus. The gold-standard POS tags for each word *w* are derived automatically from the parse trees in which *w* appears. To create the Bengali test set, we randomly chose 5K words from the vocabulary that appear at least five times in Prothom Alo. Each word in the test set was then labeled with its POS tags by two of our linguists.

Evaluation metric. Following Schütze (1995), we report performance in terms of recall, precision, and F1. Recall is the percentage of POS tags correctly proposed, precision is the percentage of POS tags proposed that are correct, and F1 is simply the harmonic mean of recall and precision. To exemplify, suppose the correct tagset for "crowd" is {NN, VB}; if our system outputs {VB, JJ, RB}, then recall is 50%, precision is 33%, and F1 is 40%. Importantly, all of our results will be reported on *word types*. This prevents the frequently occurring words from having a higher influence on the results than their infrequent counterparts.

7.2 Results and Discussion

The baseline system. We use as our baseline system one of the best existing unsupervised POS induction algorithms (Clark, 2003). More specifically, we downloaded from Clark's website[13] the code that implements a set of POS induction algorithms he proposed. Among these implementations, we chose *cluster_neyessenmorph*, which combines morphological and distributional infor-

[11] We decided to stop the bootstrapping procedure at thresholds of 0.5 and -0.5, because the more bootstrapping iterations we use, the lower are the quality of the bootstrapped data as well as the accuracy of the bootstrapped classifier.

[12] As in purification, each pairwise classifier is implemented as a set of 10 classifiers, each of which is trained on an equal number of instances from both classes. Testing also proceeds as before: the label of an instance is derived from the average of the confidence values returned by the 10 classifiers, and the confidence value associated with the label is just the average of the 10 confidence values.

[13] http://www.cs.rhul.ac.uk/home/alexc/

mation and achieves the best performance in his paper. When running his program, we use WSJ and Prothom Alo as the input corpora. In addition, we set the number of clusters produced to be 128, since this setting yields the best result in his paper. Results of the baseline system for the English and Bengali test sets are shown under the "After Bootstrapping" column in row 1 of Tables 3 and 4. As we can see, the baseline achieves F1-scores of 59% and 45% for English and Bengali, respectively. The other results in row 1 will be discussed below.

Our induction system. Recall that our unsupervised POS induction algorithm operates in three steps. To better understand the performance contribution of each of these steps, we show in row 2 of Tables 3 and 4 the results of our system after we (1) morphologically cluster the words, (2) purify the seed set, and (3) augment the seed set. Importantly, the numbers shown for each step are computed over the set of words in the test set that are labeled at the end of that step. For instance, the morphological clustering algorithm labeled 11K English words and 25K Bengali words, and so recall, precision and F1-score are computed over the subset of these labeled words that appear in the test set. Similarly, after bootstrapping, all the words that appear at least five times in our corpus are labeled; since our labeled data is now a superset of our test data, the numbers in the last column are the results of our algorithm for the entire test set.

As we can see, after morphological clustering, our system achieves F1-scores of 79% and 78% for English and Bengali, respectively. When measured on exactly the same set of words, the baseline only achieves F-scores of 59% and 56%. In fact, comparing rows 1 and 2, we outperform the baseline in each of the three steps of our algorithm. In particular, our system yields F1-scores of 73% and 77% for the entire English and Bengali test sets, thus outperforming the baseline by 14% and 18% for English and Bengali, respectively.

Two additional points deserve mentioning. First, for both languages, the highest F1-score is achieved after the purification step. A closer analysis of the labeled words reveals the reason. For English, many of the nouns incorrectly labeled as verbs by the morphological clustering algorithm were subsequently removed during the purification step when distributional similarity was used on top of morphological similarity. For Bengali, many proper nouns were assigned by the morphological clustering algorithm to the clusters dominated by common nouns (because the two types of Bengali nouns are morphologically similar), and many of these mislabeled proper nouns were subsequently removed during purification. Second, as expected, precision drops after the seed augmentation step, since the quality of the labeled data deteriorates as bootstrapping progresses. Nevertheless, with a lot more words labeled in the bootstrapping step, we still achieve F1-scores of 73% for English and 76% for Bengali.

The remaining rows of the Tables 3 and 4 show the performance of our algorithm for each tag in our two POS tagsets. Different observations can be made for the two languages. For English, the poor results for VBZ and NNS can be attributed to the fact that it is not easy to distinguish between these two tags: "s" is a typical suffix for words that are NNS and words that are the third person singular of a verb. In addition, results for verbs are better than those for nouns, since verbs are easier to identify using only morphological knowledge.

For Bengali, results for adjectives are not good, since (1) adjectives and nouns have very similar distributional property in Bengali and (2) there are not enough suffixes to induce the adjectives morphologically. Moreover, we achieve high precision but low recall for proper nouns. This implies that most of the words that our algorithm labels as proper nouns are indeed correct, but there are also many proper nouns that are mislabeled. A closer examination of the clusters reveals that many of these proper nouns are mislabeled as common nouns, presumably because these two types of Bengali nouns are morphologically and distributionally similar and therefore it is difficult to separate them. We will leave the identification of Bengali proper nouns as a topic for future research.

7.3 Additional Experiments

Labeling rare words with morphological information. Although our discussion thus far has focused on words whose corpus frequency is at least five, it would be informative to examine how well our algorithm performs on *rare, distributionally unreliable* words (i.e., words with corpus frequency less than five). Recall that our morphological clustering algorithm also clusters rare words. In fact, these rare words comprise 15% of the English words and 18% of the Bengali words in our morphological formed clusters. Perhaps more impor-

	After Morphological Clustering			After Purification			After Bootstrapping		
	P	R	F1	P	R	F1	P	R	F1
Baseline	84.1	45.3	58.9	84.9	51.4	64.1	75.6	48.0	59.0
Ours	85.9	74.0	**79.4**	89.3	74.4	**81.7**	80.4	66.8	**73.1**
JJ	88.7	49.1	63.2	91.4	51.9	66.1	57.7	62.9	60.2
JJR	91.1	86.2	88.6	92.1	92.0	92.0	62.1	83.1	71.0
JJS	100	98.3	99.1	100	100	100	81.3	86.9	83.9
NN	91.6	43.7	59.2	94.8	42.8	58.8	95.2	47.1	62.8
NNS	90.6	39.2	53.5	93.5	41.3	57.2	96.6	44.7	60.9
RB	100	76.1	86.4	100	82.2	90.6	98.8	63.5	77.3
VB	74.0	97.7	84.1	79.8	96.0	87.1	65.7	92.8	76.9
VBD	96.6	98.9	97.7	97.6	100	98.8	96.7	91.9	93.3
VBG	89.9	100	94.7	91.1	100	95.7	90.8	93.5	92.1
VBZ	60.9	99.9	74.7	65.1	96.8	77.7	52.8	92.6	67.3

Table 3: POS induction results for English based on word type

	After Morphological Clustering			After Purification			After Bootstrapping		
	P	R	F1	P	R	F1	P	R	F1
Baseline	82.1	42.3	55.5	83.1	45.3	58.3	78.1	43.3	49.3
Ours	74.1	81.3	**77.5**	83.4	78.0	**80.7**	74.1	79.2	**76.6**
JJ	50.0	51.8	50.9	56.1	55.0	55.5	57.5	51.4	54.3
NN	63.0	96.8	76.4	67.0	96.0	78.9	62.2	92.2	74.3
NN2	96.3	100	98.1	99.0	100	99.5	99.0	99.0	99.0
NN6	95.5	89.2	92.2	97.2	90.0	93.9	97.1	91.0	93.9
NN7	88.4	94.1	89.7	92.1	99.2	93.1	90.1	78.7	84.1
NNP	87.2	37.3	52.3	92.8	43.8	59.4	92.7	51.5	66.1
NNS	62.7	93.1	75.0	66.8	93.5	77.9	65.2	94.1	77.1
NNSH	91.0	100	95.6	91.0	100	95.7	91.0	100	95.7
VB	68.9	93.0	79.2	77.0	94.6	84.9	73.9	91.8	81.9
VBN	84.3	49.1	62.1	82.4	50.1	62.9	56.1	46.7	50.1

Table 4: POS induction results for Bengali based on word type

tantly, when measuring performance on just these morphologically clustered rare words, our algorithm achieves F1-scores of 81% and 79% for English and Bengali, respectively. These results provide empirical support for the claim that morphological information can be usefully employed to label rare words (Clark, 2003).

Soft clustering. Many words have more than one POS tag. For instance, "received" can be labeled as VBD and JJ. Although our morphological clustering algorithm can predict some of these ambiguities, those are at the "big" cluster level. At the sub-cluster level, the algorithm imposes a hard clustering on the words. In other words, no word appears in more than one sub-cluster.

Ideally, a POS induction algorithm should produce soft clusters due to lexical ambiguity. In fact, Jardino and Adda (1994), Schütze (1997) and Clark (2000) have attempted to address the ambiguity problem to a certain extent. We have also experimented with a very simple method for handling ambiguity in our bootstrapping algorithm: when augmenting the seed set, instead of labeling a word with a tag that receives 9 votes from the 45 pairwise classifiers, we label a word with any tag that receives at least 8 votes, effectively allowing the assignment of more than one label to a word. However, our experimental results (not shown due to space limitations) indicate that the incorporation of this method does not yield better overall performance, since many of the additional labels are erroneous and hence their presence deteriorates the quality of the bootstrapped data.

8 Conclusions

We have proposed a new bootstrapping algorithm for unsupervised POS induction. In contrast to existing algorithms developed for this problem, our algorithm is designed to (1) operate under a resource-scarce setting in which no language-specific tools or resources are available and (2) more tightly integrate morphological information with the distributional POS induction framework. In particular, our algorithm (1) improves the quality of the seed clusters by employing seed words that are distributionally and morphologically reliable and (2) uses support vector learning to combine morphological and distributional information. Our results show that it outperforms Clark's algorithm for English and Bengali, suggesting that it is applicable to both morphologically impoverished and highly inflectional languages.

Acknowledgements

We thank the five anonymous EMNLP-CoNLL referees for their valuable comments. We also thank Zeeshan Abedin and Mahbubur Rahman Haque for creating the Bengali lexicon.

References

Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*.

Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24(2):123-140.

Alexander Clark. 2000. Inducing syntactic categories by context distributional clustering. In *Proceedings of CoNLL*, pages 91-94.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the EACL*.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. In *Computer and Information Science, Report A81*, Helsinki University of Technology.

Silviu Cucerzan and David Yarowsky. 2000. Language independent, minimally supervised induction of lexical probabilities. In *Proceedings of the ACL*, pages 270-277.

Sajib Dasgupta and Vincent Ng. 2007. High-performance, language-independent morphological segmentation. In *Proceedings of NAACL-HLT*, pages 155-163.

Kevin Duh and Katrin Kirchhoff. 2006. Lexicon acquisition for dialectal Arabic using transductive learning. In *Proceedings of EMNLP*, pages 399-407.

Dayne Freitag. 2004. Toward unsupervised whole-corpus tagging. In *Proceedings of COLING*, pages 357-363.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. In *Computational Linguistics* 27(2):153-198.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of HLT-NAACL*, pages 320-327.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of the NAACL*, pages 94-101.

Zellig Harris. 1954. Distributional structure. In *Word*, 10(2/3):146-162.

Michele Jardino and Gilles Adda. 1994. Automatic determination of a stochastic bi-gram class language model. In *Proceedings of Grammatical Inference and Applications, Second International Colloquium, ICGI-94*, pages 57-65.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*, pages 44-56. MIT Press.

Sydney Lamb. 1961. On the mechanization of syntactic analysis. In *Proceedings of the 1961 Conference on Machine Translation of Languages and Applied Language Analysis*, Volume 2, pages 674-685. HMSO, London.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313-330.

Andrei Mikheev. 1997. Automatic rule induction for unknown word-guessing. *Computational Linguistics*, 23(3):405-423.

Goutam Kumar Saha, Amiya Baran Saha, and Sudipto Debnath. 2004. Computer assisted Bangla words POS tagging. In *Proceedings of the International Symposium on Machine Translation NLP and TSS (iTRANS, 2004)*.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the EACL*, pages 141-148.

Hinrich Schütze. 1997. *Ambiguity Resolution in Language Learning*. CSLI Publications.

Noah Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the ACL*, pages 354-362.

Semi-Supervised Classification for Extracting Protein Interaction Sentences using Dependency Parsing

Güneş Erkan
University of Michigan
gerkan@umich.edu

Arzucan Özgür
University of Michigan
ozgur@umich.edu

Dragomir R. Radev
University of Michigan
radev@umich.edu

Abstract

We introduce a relation extraction method to identify the sentences in biomedical text that indicate an interaction among the protein names mentioned. Our approach is based on the analysis of the paths between two protein names in the dependency parse trees of the sentences. Given two dependency trees, we define two separate similarity functions (kernels) based on cosine similarity and edit distance among the paths between the protein names. Using these similarity functions, we investigate the performances of two classes of learning algorithms, Support Vector Machines and k-nearest-neighbor, and the semi-supervised counterparts of these algorithms, transductive SVMs and harmonic functions, respectively. Significant improvement over the previous results in the literature is reported as well as a new benchmark dataset is introduced. Semi-supervised algorithms perform better than their supervised version by a wide margin especially when the amount of labeled data is limited.

1 Introduction

Protein-protein interactions play an important role in vital biological processes such as metabolic and signaling pathways, cell cycle control, and DNA replication and transcription (Phizicky and Fields, 1995). A number of (mostly manually curated) databases such as MINT (Zanzoni et al., 2002), BIND (Bader et al., 2003), and SwissProt (Bairoch and Apweiler, 2000) have been created to store protein interaction information in structured and standard formats. However, the amount of biomedical literature regarding protein interactions is increasing rapidly and it is difficult for interaction database curators to detect and curate protein interaction information manually. Thus, most of the protein interaction information remains hidden in the text of the papers in the biomedical literature. Therefore, the development of information extraction and text mining techniques for automatic extraction of protein interaction information from free texts has become an important research area.

In this paper, we introduce an information extraction approach to identify sentences in text that indicate an interaction relation between two proteins. Our method is different than most of the previous studies (see Section 2) on this problem in two aspects: First, we generate the dependency parses of the sentences that we analyze, making use of the dependency relationships among the words. This enables us to make more syntax-aware inferences about the roles of the proteins in a sentence compared to the classical pattern-matching information extraction methods. Second, we investigate semi-supervised machine learning methods on top of the dependency features we generate. Although there have been a number of learning-based studies in this domain, our methods are the first semi-supervised efforts to our knowledge. The high cost of labeling free text for this problem makes semi-supervised methods particularly valuable.

We focus on two semi-supervised learning methods: transductive SVMs (TSVM) (Joachims, 1999),

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 228–237, Prague, June 2007. ©2007 Association for Computational Linguistics

and harmonic functions (Zhu et al., 2003). We also compare these two methods with their supervised counterparts, namely SVMs and k-nearest neighbor algorithm. Because of the nature of these algorithms, we propose two similarity functions (*kernels* in SVM terminology) among the instances of the learning problem. The instances in this problem are natural language sentences with protein names in them, and the similarity functions are defined on the positions of the protein names in the corresponding parse trees. Our motivating assumption is that the *path* between two protein names in a dependency tree is a good description of the semantic relation between them in the corresponding sentence. We consider two similarity functions; one based on the cosine similarity and the other based on the edit distance among such paths.

2 Related Work

There have been many approaches to extract protein interactions from free text. One of them is based on matching pre-specified patterns and rules (Blaschke et al., 1999; Ono et al., 2001). However, complex cases that are not covered by the pre-defined patterns and rules cannot be extracted by these methods. Huang *et al.* (2004) proposed a method where patterns are discovered automatically from a set of sentences by dynamic programming. Bunescu *et al.* (2005) have studied the performance of rule learning algorithms. They propose two methods for protein interaction extraction. One is based on the rule learning method Rapier and the other on longest common subsequences. They show that these methods outperform hand-written rules.

Another class of approaches is using more syntax-aware natural language processing (NLP) techniques. Both full and partial (shallow) parsing strategies have been applied in the literature. In partial parsing the sentence structure is decomposed partially and local dependencies between certain phrasal components are extracted. An example of the application of this method is relational parsing for the *inhibition* relation (Pustejovsky et al., 2002). In full parsing, however, the full sentence structure is taken into account. Temkin and Gilder (2003) used a full parser with a lexical analyzer and a context free grammar (CFG) to extract protein-protein interaction from text. Another study that uses full-sentence parsing to extract human protein interactions is (Daraselia et al., 2004). Alternatively, Yakushiji *et al.* (2005) propose a system based on head-driven phrase structure grammar (HPSG). In their system protein interaction expressions are presented as predicate argument structure patterns from the HPSG parser. These parsing approaches consider only syntactic properties of the sentences and do not take into account semantic properties. Thus, although they are complicated and require many resources, their performance is not satisfactory.

Machine learning techniques for extracting protein interaction information have gained interest in the recent years. The PreBIND system uses SVM to identify the existence of protein interactions in abstracts and uses this type of information to enhance manual expert reviewing for the BIND database (Donaldson et al., 2003). Words and word bigrams are used as binary features. This system is also tested with the Naive Bayes classifier, but SVM is reported to perform better. Mitsumori *et al.* (2006) also use SVM to extract protein-protein interactions. They use bag-of-words features, specifically the words around the protein names. These systems do not use any syntactic or semantic information. Sugiyama *et al.* (2003) extract features from the sentences based on the verbs and nouns in the sentences such as the verbal forms, and the part of speech tags of the 20 words surrounding the verb (10 before and 10 after it). Further features are used to indicate whether a noun is found, as well as the part of speech tags for the 20 words surrounding the noun, and whether the noun contains numerical characters, non-alpha characters, or uppercase letters. They construct k-nearest neighbor, decision tree, neural network, and SVM classifiers by using these features. They report that the SVM classifier performs the best. They use part-of-speech information, but do not consider any dependency or semantic information.

The paper is organized as follows. In Section 3 we describe our method of extracting features from the dependency parse trees of the sentences and defining the similarity between two sentences. In Section 4 we discuss our supervised and semi-supervised methods. In Section 5 we describe the data sets and evaluation metrics that we used, and present our results. We conclude in Section 6.

3 Sentence Similarity Based on Dependency Parsing

In order to apply the semi-supervised harmonic functions and its supervised counterpart kNN, and the kernel based TSVM and SVM methods, we need to define a similarity measure between two sentences. For this purpose, we use the dependency parse trees of the sentences. Unlike a syntactic parse (which describes the syntactic constituent structure of a sentence), the dependency parse of a sentence captures the semantic predicate-argument relationships among its words. The idea of using dependency parse trees for relation extraction in general was studied by Bunescu and Mooney (2005a). To extract the relationship between two entities, they design a kernel function that uses the shortest path in the dependency tree between them. The motivation is based on the observation that the shortest path between the entities usually captures the necessary information to identify their relationship. They show that their approach outperforms the dependency tree kernel of Culotta and Sorensen (2004), which is based on the subtree that contains the two entities. We adapt the idea of Bunescu and Mooney (2005a) to the task of identifying protein-protein interaction sentences. We define the similarity between two sentences based on the paths between two proteins in the dependency parse trees of the sentences.

In this study we assume that the protein names have already been annotated and focus instead on the task of extracting protein-protein interaction sentences for a given protein pair. We parse the sentences with the Stanford Parser[1] (de Marneffe et al., 2006). From the dependency parse trees of each sentence we extract the shortest path between a protein pair.

For example, Figure 1 shows the dependency tree we got for the sentence "*The results demonstrated that KaiC interacts rhythmically with KaiA, KaiB, and SasA.*" This example sentence illustrates that the dependency path between a protein pair captures the relevant information regarding the relationship between the proteins better compared to using the words in the unparsed sentence. Consider the protein pair KaiC and SasA. The words in the sentence between these proteins are *interacts*, *rhythmically*, *with*, *KaiA*, *KaiB*, and *and*. Among these words *rhythmically*, *KaiA*, *and* and *KaiB* are not directly related to the interaction relationship between KaiC and SasA. On the other hand, the words in the dependency path between this protein pair give sufficient information to identify their relationship.

In this sentence we have four proteins (KaiC, KaiA, KaiB, and SasA). So there are six pairs of proteins for which a sentence may or may not be describing an interaction. The following are the paths between the six protein pairs. In this example there is a single path between each protein pair. However, there may be more than one paths between a protein pair, if one or both appear multiple times in the sentence. In such cases, we select the shortest paths between the protein pairs.

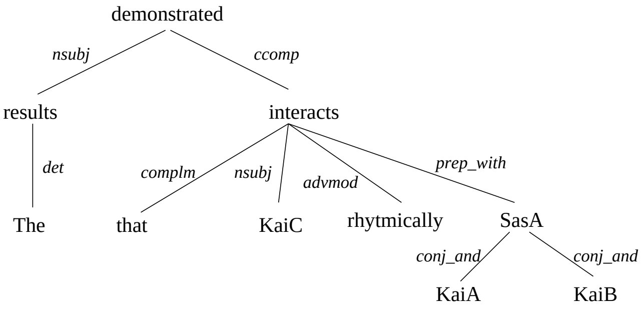


Figure 1: The dependency tree of the sentence "*The results demonstrated that KaiC interacts rhythmically with KaiA, KaiB, and SasA.*"

1. KaiC - nsubj - interacts - prep_with - SasA
2. KaiC - nsubj - interacts - prep_with - SasA - conj_and - KaiA
3. KaiC - nsubj - interacts - prep_with – SasA - conj_and - KaiB
4. SasA - conj_and - KaiA
5. SasA - conj_and - KaiB
6. KaiA – conj_and – SasA - conj_and - KaiB

If a sentence contains n different proteins, there are $\binom{n}{2}$ different pairs of proteins. We use machine learning approaches to classify each sentence as an interaction sentence or not for a protein pair. A sentence may be an interaction sentence for one protein

[1] http://nlp.stanford.edu/software/lex-parser.shtml

pair, while not for another protein pair. For instance, our example sentence is a positive interaction sentence for the *KaiC* and *SasA* protein pair. However, it is a negative interaction sentence for the *KaiA* and *SasA* protein pair, i.e., it does not describe an interaction between this pair of proteins. Thus, before parsing a sentence, we make multiple copies of it, one for each protein pair. To reduce data sparseness, we rename the proteins in the pair as *PROTX1* and *PROTX2*, and all the other proteins in the sentence as *PROTX0*. So, for our example sentence we have the following instances in the training set:

1. *PROTX1* - nsubj - interacts - prep_with - *PROTX2*
2. *PROTX1* - nsubj - interacts - prep_with - *PROTX0* - conj_and - *PROTX2*
3. *PROTX1* - nsubj - interacts - prep_with – *PROTX0* - conj_and - *PROTX2*
4. *PROTX1* - conj_and - *PROTX2*
5. *PROTX1* - conj_and - *PROTX2*
6. *PROTX1* – conj_and – *PROTX0* - conj_and - *PROTX2*

The first three instances are positive as they describe an interaction between *PROTX1* and *PROTX2*. The last three are negative, as they do not describe an interaction between *PROTX1* and *PROTX2*.

We define the similarity between two instances based on cosine similarity and edit distance based similarity between the paths in the instances.

3.1 Cosine Similarity

Suppose p_i and p_j are the paths between *PROTX1* and *PROTX2* in instance x_i and instance x_j, respectively. We represent p_i and p_j as vectors of term frequencies in the vector-space model. The cosine similarity measure is the cosine of the angle between these two vectors and is calculated as follows:

$$cos\_sim(p_i, p_j) = cos(\mathbf{p_i}, \mathbf{p_j}) = \frac{\mathbf{p_i} \bullet \mathbf{p_j}}{\|\mathbf{p_i}\|\|\mathbf{p_j}\|} \quad (1)$$

that is, it is the dot product of $\mathbf{p_i}$ and $\mathbf{p_j}$ divided by the lengths of $\mathbf{p_i}$ and $\mathbf{p_j}$. The cosine similarity measure takes values in the range $[0, 1]$. If all the terms in p_i and p_j are common, then it takes the maximum value of 1. If none of the terms are common, then it takes the minimum value of 0.

3.2 Similarity Based on Edit Distance

A shortcoming of cosine similarity is that it only takes into account the common terms, but does not consider their order in the path. For this reason, we also use a similarity measure based on edit distance (also called Levenshtein distance). Edit distance between two strings is the minimum number of operations that have to be performed to transform the first string to the second. In the original character-based edit distance there are three types of operations. These are insertion, deletion, or substitution of a single character. We modify the character-based edit distance into a word-based one, where the operations are defined as insertion, deletion, or substitution of a single word.

The edit distance between path 1 and path 2 of our example sentence is 2. We insert *PROTX0* and *conj_and* to path 1 to convert it to path 2.

1. *PROTX1* - nsubj - interacts - prep_with - **insert (*PROTX0*)** - **insert (*conj_and*)** – *PROTX2*
2. *PROTX1* - nsubj - interacts - prep_with - *PROTX0* - conj_and - *PROTX2*

We normalize edit distance by dividing it by the length (number of words) of the longer path, so that it takes values in the range $[0, 1]$. We convert the distance measure into a similarity measure as follows.

$$edit\_sim(p_i, p_j) = e^{-\gamma(edit\_distance(p_i, p_j))} \quad (2)$$

Bunescu and Mooney (2005a) propose a similar method for relation extraction in general. However, their similarity measure is based on the number of the overlapping words between two paths. When two paths have different lengths, they assume the similarity between them is zero. On the other hand, our edit distance based measure can also account for deletions and insertions of words.

4 Semi-Supervised Machine Learning Approaches

4.1 kNN and Harmonic Functions

When a similarity measure is defined among the instances of a learning problem, a simple and natural choice is to use a nearest neighbor based approach that classifies each instance by looking at the labels of the instances that are most similar to it. Perhaps the simplest and most popular similarity-based learning algorithm is the k-nearest neighbor classification method (kNN). Let U be the set of unlabeled instances, and L be the set of labeled instances in a learning problem. Given an instance $x \in U$, let $N_k^L(x)$ be the set of top k instances in L that are most similar to x with respect to some similarity measure. The kNN equation for a binary classification problem can be written as:

$$y(x) = \sum_{z \in N_k^L(x)} \frac{sim(x,z)y(z)}{\sum_{z' \in N_k^L(x)} sim(x,z')} \qquad (3)$$

where $y(z) \in \{0,1\}$ is the label of the instance z.[2] Note that $y(x)$ can take any real value in the $[0,1]$ interval. The final classification decision is made by setting a threshold in this interval (e.g. 0.5) and classifying the instances above the threshold as positive and others as negative. For our problem, each instance is a dependency path between the proteins in the pair and the similarity function can be one of the functions we have defined in Section 3.

Equation 3 can be seen as averaging the labels (0 or 1) of the nearest neighbors of each unlabeled instance. This suggests a generalized semi-supervised version of the same algorithm by incorporating unlabeled instances as neighbors as well:

$$y(x) = \sum_{z \in N_k^{L \cup U}(x)} \frac{sim(x,z)y(z)}{\sum_{z' \in N_k^{L \cup U}(x)} sim(x,z')} \qquad (4)$$

Unlike Equation 3, the unlabeled instances are also considered in Equation 4 when finding the nearest neighbors. We can visualize this as an undirected graph, where each data instance (labeled or unlabeled) is a node that is connected to its k nearest neighbor nodes. The value of $y(\cdot)$ is set to 0 or 1 for labeled nodes depending on their class. For each unlabeled node x, $y(x)$ is equal to the average of the $y(\cdot)$ values of its neighbors. Such a function that satisfies the average property on all unlabeled nodes is called a *harmonic* function and is known to exist and have a unique solution (Doyle and Snell, 1984). Harmonic functions were first introduced as a semi-supervised learning method by Zhu *et al.* (2003). There are interesting alternative interpretations of a harmonic function on a graph. One of them can be explained in terms of random walks on a graph. Consider a random walk on a graph where at each time point we move from the current node to one of its neighbors. The next node is chosen among the neighbors of the current node with probability proportional to the weight (similarity) of the edge that connects the two nodes. Assuming we start the random walk from the node x, $y(x)$ in Equation 4 is then equal to the probability that this random walk will hit a node labeled 1 before it hits a node labeled 0.

4.2 Transductive SVM

Support vector machines (SVM) is a supervised machine learning approach designed for solving two-class pattern recognition problems. The aim is to find the decision surface that separates the positive and negative labeled training examples of a class with maximum margin (Burges, 1998).

Transductive support vector machines (TSVM) are an extension of SVM, where unlabeled data is used in addition to labeled data. The aim now is to assign labels to the unlabeled data and find a decision surface that separates the positive and negative instances of the original labeled data and the (now labeled) unlabeled data with maximum margin. Intuitively, the unlabeled data pushes the decision boundary away from the dense regions. However, unlike SVM, the optimization problem now is NP-hard (Zhu, 2005). Pointers to studies for approximation algorithms can be found in (Zhu, 2005).

In Section 3 we defined the similarity between two instances based on the cosine similarity and the edit distance based similarity between the paths in the instances. Here, we use these path similarity measures as kernels for SVM and TSVM and modify the SVM^{light} package (Joachims, 1999) by plugging in our two kernel functions.

A well-defined kernel function should be symmetric positive definite. While cosine kernel is well-defined, Cortes *et al.* (2004) proved that edit kernel is not always positive definite. However, it is possible to make the kernel matrix positive definite by adjusting the γ parameter, which is a positive real number. Li and Jiang (2005) applied the edit kernel to predict initiation sites in eucaryotic mRNAs and

[2] Equation 3 is the weighted (or *soft*) version of the kNN algorithm. In the classical *voting* scheme, x is classified in the category that the majority of its neighbors belong to.

obtained improved results compared to polynomial kernel.

5 Experimental Results

5.1 Data Sets

One of the problems in the field of protein-protein interaction extraction is that different studies generally use different data sets and evaluation metrics. Thus, it is difficult to compare their results. Bunescu *et al.* (2005) manually developed the *AIMED* corpus[3] for protein-protein interaction and protein name recognition. They tagged 199 Medline abstracts, obtained from the Database of Interacting Proteins (DIP) (Xenarios et al., 2001) and known to contain protein interactions. This corpus is becoming a standard, as it has been used in the recent studies by (Bunescu et al., 2005; Bunescu and Mooney, 2005b; Bunescu and Mooney, 2006; Mitsumori et al., 2006; Yakushiji et al., 2005).

In our study we used the *AIMED* corpus and the *CB* (Christine Brun) corpus that is provided as a resource by BioCreAtIvE II (Critical Assessment for Information Extraction in Biology) challenge evaluation[4]. We pre-processed the CB corpus by first annotating the protein names in the corpus automatically and then, refining the annotation manually. As discussed in Section 3, we pre-processed both of the data sets as follows. We replicated each sentence for each different protein pair. For n different proteins in a sentence, $\binom{n}{2}$ new sentences are created, as there are that many different pairs of proteins. In each newly created sentence we marked the protein pair considered for interaction as *PROTX1* and *PROTX2*, and all the remaining proteins in the sentence as *PROTX0*. If a sentence describes an interaction between *PROTX1* and *PROTX2*, it is labeled as positive, otherwise it is labeled as negative. The summary of the data sets after pre-processing is displayed in Table 1[5].

Since previous studies that use AIMED corpus perform 10-fold cross-validation. We also performed 10-fold cross-validation in both data sets and report the average results over the runs.

Data Set	Sentences	+ Sentences	- Sentences
AIMED	4026	951	3075
CB	4056	2202	1854

Table 1: Data Sets

5.2 Evaluation Metrics

We use precision, recall, and F-score as our metrics to evaluate the performances of the methods. Precision (π) and recall (ρ) are defined as follows:

$$\pi = \frac{TP}{TP + FP}; \qquad \rho = \frac{TP}{TP + FN} \tag{5}$$

Here, TP (True Positives) is the number of sentences classified correctly as positive; FP (False Positives) is the number of negative sentences that are classified as positive incorrectly by the classifier; and FN (False Negatives) is the number of positive sentences that are classified as negative incorrectly by the classifier.
F-score is the harmonic mean of recall and precision.

$$F\text{-}score = \frac{2\pi\rho}{\pi + \rho} \tag{6}$$

5.3 Results and Discussion

We evaluate and compare the performances of the semi-supervised machine learning approaches (TSVM and harmonic functions) with their supervised counterparts (SVM and kNN) for the task of protein-protein interaction extraction. As discussed in Section 3, we use cosine similarity and edit distance based similarity as similarity functions in harmonic functions and kNN, and as kernel functions in TSVM and SVM. Our instances consist of the shortest paths between the protein pairs in the dependency parse trees of the sentences. In our experiments, we tuned the γ parameter of the edit distance based path similarity function to 4.5 with cross-validation. The results in Table 2 and Table 3 are obtained with 10-fold cross-validation. We report the average results over the runs.

Table 2 shows the results obtained for the AIMED data set. Edit distance based path similarity function performs considerably better than the cosine similarity function with harmonic functions and kNN and usually slightly better with SVM and TSVM. We achieve our best F-score performance of 59.96% with TSVM with edit kernel. While SVM with edit

[3] ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/

[4] http://biocreative.sourceforge.net/biocreative_2.html

[5] The pre-processed data sets are available at http://belobog.si.umich.edu/clair/biocreative

kernel achieves the highest precision of 77.52%, it performs slightly worse than SVM with cosine kernel in terms of F-score measure. TSVM performs slightly better than SVM, both of which perform better than harmonic functions. kNN is the worst performing algorithm for this data set.

In Table 2, we also show the results obtained previously in the literature by using the same data set. Yakushiji *et al.* (2005) use an HPSG parser to produce predicate argument structures. They utilize these structures to automatically construct protein interaction extraction rules. Mitsumori *et al.* (2006) use SVM with the unparsed text around the protein names as features to extract protein interaction sentences. Here, we show their best result obtained by using the three words to the left and to the right of the proteins. The most closely related study to ours is that by Bunescu and Mooney (2005a). They define a kernel function based on the shortest path between two entities of a relationship in the dependency parse tree of a sentence (the SPK method). They apply this method to the domain of protein-protein interaction extraction in (Bunescu and Mooney, 2006). Here, they also test the methods ELCS (Extraction Using Longest Common Subsequences) (Bunescu et al., 2005) and SSK (Subsequence Kernel) (Bunescu and Mooney, 2005b). We cannot compare our results to theirs directly, because they report their results as a precision-recall graph. However, the best F-score in their graph seems to be around 0.50 and definitely lower than the best F-scores we have achieved (≈ 0.59). Bunescu and Mooney (2006) also use SVM as their learning method in their SPK approach. They define their similarity based on the number of overlapping words between two paths and assign a similarity of zero if the two paths have different lengths. Our improved performance with SVM and the shortest path dependency features may be due to the edit-distance based kernel, which takes into account not only the overlapping words, but also word order and accounts for deletions and insertions of words. Our results show that, SVM, TSVM, and harmonic functions achieve better F-score and recall performances than the previous studies by Yakushiji *et al.* (2005), Mitsumori *et al.* (2006), and the SSK and ELCS approaches of Bunescu and Mooney (2006). SVM and TSVM also achieve higher precision scores. Since, Mitsumori *et al.* (2006) also use SVM in their study, our improved results with SVM confirms our motivation of using dependency paths as features.

Table 3 shows the results we got with the CB data set. The F-score performance with the edit distance based similarity function is always better than that of cosine similarity function for this data set. The difference in performances is considerable for harmonic functions and kNN. Our best F-score is achieved with TSVM with edit kernel (85.22%). TSVM performs slightly better than SVM. When cosine similarity function is used, kNN performs better than harmonic functions. However, when edit distance based similarity is used, harmonic functions achieve better performance. SVM and TSVM perform better than harmonic functions. But, the gap in performance is low when edit distance based similarity is used with harmonic functions.

Method	Precision	Recall	F-Score
SVM-edit	77.52	43.51	55.61
SVM-cos	61.99	54.99	58.09
TSVM-edit	59.59	60.68	59.96
TSVM-cos	58.37	61.19	59.62
Harmonic-edit	44.17	74.20	55.29
Harmonic-cos	36.02	67.65	46.97
kNN-edit	68.77	42.17	52.20
kNN-cos	40.37	49.49	44.36
(Yakushiji et al., 2005)	33.70	33.10	33.40
(Mitsumori et al., 2006)	54.20	42.60	47.70

Table 2: Experimental Results – AIMED Data Set

Method	Precision	Recall	F-Score
SVM-edit	85.15	84.79	84.96
SVM-cos	87.83	81.45	84.49
TSVM-edit	85.62	84.89	85.22
TSVM-cos	85.67	84.31	84.96
Harmonic-edit	86.69	80.15	83.26
Harmonic-cos	72.28	70.91	71.56
kNN-edit	72.89	86.95	79.28
kNN-cos	65.42	89.49	75.54

Table 3: Experimental Results – CB Data Set

Semi-supervised approaches are usually more effective when there is less labeled data than unlabeled data, which is usually the case in real applications. To see the effect of semi-supervised approaches we perform experiments by varying the amount of la-

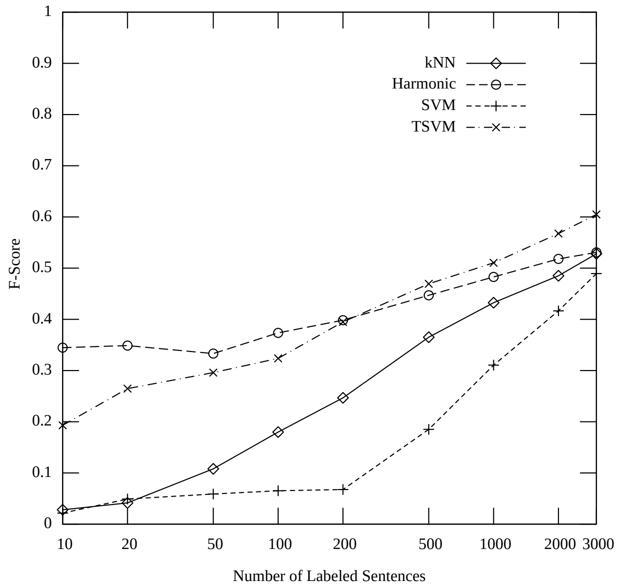


Figure 2: The F-score on the AIMED dataset with varying sizes of training data

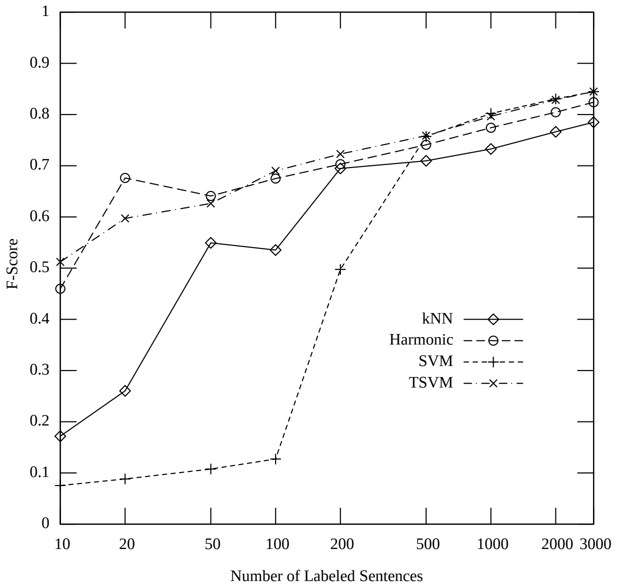


Figure 3: The F-score on the CB dataset with varying sizes of training data

beled training sentences in the range $[10, 3000]$. For each labeled training set size, sentences are selected randomly among all the sentences, and the remaining sentences are used as the unlabeled test set. The results that we report are the averages over 10 such random runs for each labeled training set size. We report the results for the algorithms when edit distance based similarity is used, as it mostly performs better than cosine similarity. Figure 2 shows the results obtained over the AIMED data set. Semi-supervised approaches TSVM and harmonic functions perform considerably better than their supervised counterparts SVM and kNN when we have small number of labeled training data. It is interesting to note that, although SVM is one of the best performing algorithms with more training data, it is the worst performing algorithm with small amount of labeled training sentences. Its performance starts to increase when number of training data is larger than 200. Eventually, its performance gets close to that of the other algorithms. Harmonic functions is the best performing algorithm when we have less than 200 labeled training data. TSVM achieves better performance when there are more than 500 labeled training sentences.

Figure 3 shows the results obtained over the CB data set. When we have less than 500 labeled sentences, harmonic functions and TSVM perform significantly better than kNN, while SVM is the worst performing algorithm. When we have more than 500 labeled training sentences, kNN is the worst performing algorithm, while the performance of SVM increases and gets similar to that of TSVM and slightly better than that of harmonic functions.

6 Conclusion

We introduced a relation extraction approach based on dependency parsing and machine learning to identify protein interaction sentences in biomedical text. Unlike syntactic parsing, dependency parsing captures the semantic predicate argument relationships between the entities in addition to the syntactic relationships. We extracted the shortest paths between protein pairs in the dependency parse trees of the sentences and defined similarity functions (kernels in SVM terminology) for these paths based on cosine similarity and edit distance. Supervised machine learning approaches have been applied to this domain. However, they rely only on labeled training data, which is difficult to gather. To our knowledge, this is the first effort in this domain to apply semi-supervised algorithms, which make use of both labeled and unlabeled data. We evaluated and compared the performances of two semi-supervised machine learning approaches (harmonic functions and TSVM), with their supervised counterparts (kNN and SVM). We showed that, edit distance based similarity function performs better than cosine similarity function since it takes into account not only common words, but also word order. Our 10-fold cross validation results showed that, TSVM performs slightly better than SVM, both of which perform better than harmonic functions. The worst performing algorithm is kNN. We compared our results with previous results published with the AIMED data set. We achieved the best F-score performance with TSVM with the edit distance kernel (59.96%) which is significantly higher than the previously reported results for the same data set.

In most real-world applications there are much more unlabeled data than labeled data. Semi-supervised approaches are usually more effective in these cases, because they make use of both the labeled and unlabeled instances when making decisions. To test this hypothesis for the application of extracting protein interaction sentences from text, we performed experiments by varying the number of labeled training sentences. Our results show that, semi-supervised algorithms perform considerably better than their supervised counterparts, when there are small number of labeled training sentences. An interesting result is that, in such cases SVM performs significantly worse than the other algorithms. Harmonic functions achieve the best performance when there are only a few labeled training sentences. As number of labeled training sentences increases the performance gap between supervised and semi-supervised algorithms decreases.

Acknowledgments

This work was supported in part by grants R01-LM008106 and U54-DA021519 from the US National Institutes of Health.

References

G. Bader, D. Betel, and C. Hogue. 2003. Bind - the biomolecular interaction network database. *Nucleic Acids Research*, 31(1):248–250.

A. Bairoch and R. Apweiler. 2000. The swiss-prot protein sequence database and its supplement trembl in 2000. *Nucleic Acids Research*, 28(1):45–48.

C. Blaschke, M. A. Andrade, C. A. Ouzounis, and A. Valencia. 1999. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Proceedings of the AAAI Conference on Intelligent Systems for Molecular Biology (ISMB 1999)*, pages 60–67.

R. C. Bunescu and R. J. Mooney. 2005a. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, B.C, October.

R. C. Bunescu and R. J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, Vancouver, B.C, December.

R. C. Bunescu and R. J. Mooney, 2006. *Text Mining and Natural Language Processing*, chapter Extracting Relations from Text: From Word Sequences to Dependency Paths. forthcoming book.

R. Bunescu, R. Ge, J. R. Kate, M. E. Marcotte, R. J. Mooney, K. A. Ramani, and W. Y. Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155, February.

C. J. C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

C. Cortes, P. Haffner, and M. Mohri. 2004. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, (5):1035–1062, August.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain, July.

N. Daraselia, A. Yuryev, S. Egorov, S. Novichkova, A. Nikitin, and I. Mazo. 2004. Extracting human protein interactions from medline using a full-sentence parser. *Bioinformatics*, 20(5):604–611.

M-C. de Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the IEEE / ACL 2006 Workshop on Spoken Language Technology*. The Stanford Natural Language Processing Group.

I. Donaldson, J. Martin, B. de Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G. D. Bader, K. Michalockova, T. Pawson, and C. W. V. Hogue. 2003. Prebind and textomy - mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, 4:11.

P. G. Doyle and J. L. Snell. 1984. *Random Walks and Electric Networks*. Mathematical Association of America.

M. Huang, X. Zhu, Y. Hao, D. G. Payan, K. Qu, and M. Li. 2004. Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20(18):3604–3612.

T. Joachims. 1999. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers, San Francisco, US.

H. Li and T. Jiang. 2005. A class of edit kernels for svms to predict translation initiation sites in eukaryotic mrnas. *Journal of Computational Biology*, 12(6):702–718.

T. Mitsumori, M. Murata, Y. Fukuda, K. Doi, and H. Doi. 2006. Extracting protein-protein interaction information from biomedical text with svm. *IEICE Transactions on Information and Systems*, E89-D(8):2464–2466.

T. Ono, H. Hishigaki, A. Tanigami, and T. Takagi. 2001. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17(2):155–161.

E. M. Phizicky and S. Fields. 1995. Protein-protein interactions: methods for detection and analysis. *Microbiol. Rev.*, 59(1):94–123, March.

J. Pustejovsky, J. Castano, J. Zhang, M. Kotecki, and B. Cochran. 2002. Robust relational parsing over biomedical literature: Extracting inhibit relations. In *Proceedings of the seventh Pacific Symposium on Biocomputing (PSB 2002)*, pages 362–373.

K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. 2003. Extracting information on protein-protein interactions from biological literature based on machine learning approaches. *Genome Informatics*, 14:699–700.

J. M. Temkin and M. R. Gilder. 2003. Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, 19:2046–2053.

I. Xenarios, E. Fernandez, L. Salwinski, X. J. Duan, M. J. Thompson, E. M. Marcotte, and D. Eisenberg. 2001. Dip: The database of interacting proteins: 2001 update. *Nucleic Acids Res.*, 29:239 – 241, January.

A. Yakushiji, Y. Miyao, Y. Tateisi, and J. Tsujii. 2005. Biomedical information extraction with predicate-argument structure patterns. In *Proceedings of The Eleventh Annual Meeting of The Association for Natural Language Processing*, pages 93–96.

A. Zanzoni, L. Montecchi-Palazzi, M. Quondam, G. Ausiello, M. Helmer-Citterich, and G. Cesareni. 2002. Mint: A molecular interaction database. *FEBS Letters*, 513:135–140.

X. Zhu, Z. Ghahramani, and J. D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In T. Fawcett and N. Mishra, editors, *ICML*, pages 912–919. AAAI Press.

X. Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.

A Sequence Alignment Model Based on the Averaged Perceptron

Dayne Freitag
Fair Isaac Corporation
3661 Valley Centre Drive
San Diego, CA 92130, USA
DayneFreitag@fairisaac.com

Shahram Khadivi
Lehrstuhl für Informatik 6
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
khadivi@cs.rwth-aachen.de

Abstract

We describe a discriminatively trained sequence alignment model based on the averaged perceptron. In common with other approaches to sequence modeling using perceptrons, and in contrast with comparable generative models, this model permits and transparently exploits arbitrary features of input strings. The simplicity of perceptron training lends more versatility than comparable approaches, allowing the model to be applied to a variety of problem types for which a learned edit model might be useful. We enumerate some of these problem types, describe a training procedure for each, and evaluate the model's performance on several problems. We show that the proposed model performs at least as well as an approach based on statistical machine translation on two problems of name transliteration, and provide evidence that the combination of the two approaches promises further improvement.

1 Introduction

Sequence alignment is a problem that crops up in many forms, both in computational linguistics (CL) and in other endeavors. The ability to find an optimal alignment between two sequences has found application in a number of areas of CL, including phonetic modeling (Ristad and Yianilos, 1998), name transcription (Huang et al., 2004), and duplicate detection or information integration (Bilenko and Mooney, 2003; McCallum et al., 2005). Sequence alignment is a member of a broader class of problems which we might call *sequence transduction*, to which one of the core CL challenges, machine translation, belongs.

Under the assumption that one string (the *target*) is produced through a series of local edits to another string (the *source*), and given an edit cost matrix, the optimal sequence of edits can be efficiently computed through dynamic programming (Needleman and Wunsch, 1970). While the cost matrix traditionally has been set by hand, several recent papers have proposed determining edit costs empirically. These proposals arise from a variety of learning paradigms, including generative models (Ristad and Yianilos, 1998; Bilenko and Mooney, 2003), conditional random fields (McCallum et al., 2005), maximum-margin methods (Joachims, 2003), and gradient boosting (Parker et al., 2006). While approaches based on generative models support only limited feature engineering, discriminative approaches share the advantage of allowing arbitrary features of the input sequences.

We describe a new sequence alignment model based on the averaged perceptron (Collins, 2002), which shares with the above approaches the ability to exploit arbitrary features of the input sequences, but is distinguished from them by its relative simplicity and the incremental character of its training procedure. The fact that it is an online algorithm makes it straightforward to adapt to a range of problems. To show this, we evaluate the approach on several different tasks, some of them merely illustrative, but some with clear practical significance,

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 238–247, Prague, June 2007. ©2007 Association for Computational Linguistics

particularly the problem of named entity transcription.

2 The Algorithm

2.1 The Formalism

Suppose we are given two sequences, $\boldsymbol{s}_1^m \in \Sigma_s^*$ and $\boldsymbol{t}_1^n \in \Sigma_t^*$. We desire a real-valued function $A(\boldsymbol{s}, \boldsymbol{t})$ which assigns high scores to pairs $\boldsymbol{s}, \boldsymbol{t}$ with high *affinity*, where affinity is an application-specific notion (e.g., $\boldsymbol{t}$ is a likely phoneme sequence represented by the letter sequence $\boldsymbol{s}$). If we stipulate that this score is the sum of the individual scores of a series of edits, we can find the highest-scoring such series through a generalization of the standard edit distance:

$$A(\boldsymbol{s}_1^i, \boldsymbol{t}_1^j) = \max \begin{cases} a_{\epsilon,t_j}(\boldsymbol{s}, i, \boldsymbol{t}, j) + A(\boldsymbol{s}_1^i, \boldsymbol{t}_1^{j-1}) \\ a_{s_i,\epsilon}(\boldsymbol{s}, i, \boldsymbol{t}, j) + A(\boldsymbol{s}_1^{i-1}, \boldsymbol{t}_1^{j}) \\ a_{s_i,t_j}(\boldsymbol{s}, i, \boldsymbol{t}, j) + A(\boldsymbol{s}_1^{i-1}, \boldsymbol{t}_1^{j-1}) \end{cases} \quad (1)$$

with $A(\emptyset, \emptyset) = 0$. The function $a_{s_i,t_j}(\boldsymbol{s}, i, \boldsymbol{t}, j)$ represents the score of substituting t_j for s_i; a_{ϵ,t_j} and $a_{s_i,\epsilon}$ represent insertion and deletion, respectively. If we assume constant-time computation of primitive edit costs, this recursive definition of A allows us to find the highest scoring series of edits for a given sequence pair in time proportional to the product of their lengths. Note that a is indexed by the characters involved in an edit (i.e., inserting 'e' generally has a different cost than inserting 's'). Note further that the score associated with a particular operation may depend on any features computable from the respective positions in the two sequences.

In the experiments reported in this paper, we assume that each local function a is defined in terms of $p + q$ features, $\{f_1, \cdots, f_p, f_{p+1}, \cdots, f_{p+q}\}$, and that these features have the functional form $\Sigma^* \times \mathcal{N} \mapsto \mathcal{R}$. In other words, each feature takes a sequence and an index and returns a real value. The first p features are defined over sequences from the source alphabet, while the remaining q are defined over the target alphabet.[1] In this paper we use character n-gram indicator features.

```
Given a set $\boldsymbol{S}$ of source sequences
$V \leftarrow []$, an empty list
$\alpha \leftarrow \mathbf{0}$, a weight vector
for some number of iterations do
    for $\boldsymbol{s}$ in $\boldsymbol{S}$ do
        Pick $\boldsymbol{t}, \boldsymbol{t'}$, $\boldsymbol{t}$ having higher affinity with $\boldsymbol{s}$
        $\langle \boldsymbol{e}, v \rangle \leftarrow A_\alpha(\boldsymbol{s}, \boldsymbol{t})$
        $\langle \boldsymbol{e'}, v' \rangle \leftarrow A_\alpha(\boldsymbol{s}, \boldsymbol{t'})$
        if $v' \geq v$ then
            $\alpha \leftarrow \alpha + \Phi(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{e}) - \Phi(\boldsymbol{s}, \boldsymbol{t'}, \boldsymbol{e'})$
        end if
        Append $\alpha$ to $V$
    end for
end for
Return the mean $\alpha$ from $V$
```

Table 1: The training algorithm. A_α is the affinity function under model parameters α, returning edit sequence $\boldsymbol{e}$ and score v.

The score of a particular edit is a linear combination of the corresponding feature values:

$$a(\boldsymbol{s}, i, \boldsymbol{t}, j) = \sum_{k=1}^{p} \alpha_k \cdot f_k(\boldsymbol{s}, i) + \sum_{k=p+1}^{p+q} \alpha_k \cdot f_k(\boldsymbol{t}, j) \quad (2)$$

The weights α_k are what we seek to optimize in order to tune the model for our particular application.

2.2 A Perceptron-Based Edit Model

In this section we present a general-purpose extension of perceptron training for sequence labeling, due to Collins (2002), to the problem of sequence alignment. Take α to be a model parameterization, and let $A_\alpha(\boldsymbol{s}, \boldsymbol{t})$ return an optimal edit sequence $\boldsymbol{e}$, with its score v, given input sequences $\boldsymbol{s}$ and $\boldsymbol{t}$ under α. Elements of sequence $\boldsymbol{e}$ are character pairs $\langle c_s, c_t \rangle$, with $c_s \in \Sigma_s \cup \{\epsilon\}$ and $c_t \in \Sigma_t \cup \{\epsilon\}$, where ϵ represents the empty string. Let $\Phi(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{e})$ be a feature vector, having the same dimensionality as α, for a source, target, and corresponding edit sequence. This feature vector is the sum of feature vectors at each point in $\boldsymbol{e}$ as it is played out along input sequences $\boldsymbol{s}$ and $\boldsymbol{t}$.

Table 1 shows the basic algorithm. Starting with a zero parameter vector, we iterate through the collection of source sequences. For each sequence, we pick two target sequences having unequal affinity

[1] Of course, features that depend jointly on both sequences may also be of interest.

with the source sequence (Line 6). If the scores returned by our current model (Lines 7 and 8) agree with our ordering, we do nothing. Otherwise, we update the model using the perceptron training rule (Line 10). Ultimately, we return $\boldsymbol{\alpha}$ averaged over all datapoint presentations.

2.3 Training Modes

The algorithm presented in Table 1 does not specify how the two target sequences $\boldsymbol{t}$ and $\boldsymbol{t'}$ are to be chosen in Line 6. The answer to this question depends on the application. There are fundamentally two settings, depending on whether or not target strings are drawn from the same set as source strings; we will call the setting in which source and target strings inhabit the same set the *affinity* setting, and refer to the the case where they form different sets as the *transduction* setting. Here, we sketch four problem scenarios, two from each setting, and specify a target selection procedure appropriate for each.

Affinity, ranking. The task poses a latent affinity between strings, but we can measure it only indirectly. In particular, we can order *some* of the target sequences according to their affinity with a source sequence $\boldsymbol{s}$. In this case, we train as follows: Order a sample of the target sequences according to this partial order. Let $\boldsymbol{t}$ and $\boldsymbol{t'}$ be two sequences from this order, such that $\boldsymbol{t}$ is ordered higher than $\boldsymbol{t'}$.

Affinity, classification. The sequences in Σ^* can be grouped into classes, and we wish the model to assign high affinity to co-members of a class and low affinity to members of different classes. Train as follows: For each $\boldsymbol{s}$, sample $\boldsymbol{t}$ from among its co-members and $\boldsymbol{t'}$ from among the members of other classes.

Transduction, ranking. The data is presented as source-target pairs, where each $\boldsymbol{t}$ is a transduction of the corresponding $\boldsymbol{s}$. We wish to learn a model which, given a novel $\boldsymbol{s}$, will enable us to rank candidate transductions. Train as follows: Given $\boldsymbol{s}$, let $\boldsymbol{t}$ be the target sequence provided to us. Sample $\boldsymbol{t'}$ from among the other target sequences.

Transduction, generation. We are again given source-target pairs. We wish to learn to generate a probable target string, given a novel source string. Train as follows: *Generate* a $\boldsymbol{t'}$ that is approximately optimal according to the current model. Note that since edit decisions are based in part on (arbitrary) features of the target sequence, and since generation involves construction of the target sequence, it is not uncommon for a greedy generator to make edit decisions which are locally optimal, but which result several edits later in a partially constructed sequence in which no good edits are available. Thus, the problem of generation does not correspond to a simple recurrence relation like Equation 1. Consequently, we experimented with several heuristic approaches to generation and found that a beam search works well.

$\texttt{edit}_{\wedge}\texttt{ing}$	$\rightarrow$	$\texttt{STR}_{\wedge}\texttt{INGS}$
$f_{s,it\_}$		$f_{t,TR\_}$
$f_{s,t\_}$		$f_{t,R\_}$
$f_{s,\_in}$		$f_{\emptyset}$
$f_{s,\_i}$		

Table 2: Features with non-zero value for an example string pair and a model of order 2.

3 Evaluation

To establish the effectiveness of the model, we trained it on a range of problems, including instances of each of the four settings enumerated above. Problems ranged from the merely illustrative to a non-trivial application of computational linguistics.

3.1 Feature Construction

Without exception, the features we provide to the algorithm are the same in all experiments. Given a user-specified *order* k, we define a Boolean feature for every distinct character gram observed in the data of length k or smaller. Recall that there are two disjoint sets of features, those defined over strings drawn from the source and target alphabets, respectively. Given a source string and index, those features have value 1 whose corresponding grams (of size k or smaller) are observed preceding or following the index (preceding features are distinct from following ones); given a target string and index, we observe only preceding grams. Although it is possible to observe following grams in the target string in some settings, it is not possible in general (i.e., not when generating strings). We therefore adhere to this restriction for convenience and uniformity.

An example will make this clear. In Table 2 we are midway through the conversion of the source string “editing” into the target string “STRINGS”. Below the two strings are those gram features which have non-zero value at the indicated cursors. The underbar character encodes on which side of the cursor a gram is observed. Note that an empty-gram feature, which always tests true, is also included, allowing us to experiment with 0-order models.

3.2 Illustrative Problems

To test the ability of the model to recover known edit affinities, we experimented with a simple artificial problem. Using a large list of English words, we define an edit affinity that is sensitive only to consonants. Specifically, the affinity between two words is the maximum number of consonant self-substitutions, with any substitutions involving the first five consonants counting for five normal substitutions. Thus, substituting ‘b’ for ‘b’ contributes 5 to the score, substituting ‘z’ for ‘z’ contributes 1, while operations other than self-subsitutions, and any operations involving vowels, contribute 0.

One epoch of training is conducted as follows. For each word s in the training set, we choose 10 other words from the set at random and sort these words according to both the true and estimated affinity. Let t be the string with highest true affinity; let t' (the decoy) be the string with highest estimated affinity. We performed 3-fold cross-validation on a collection of 33,432 words, in each fold training the model for 5 epochs.[2]

Our performance metric is *ranking accuracy*, the fraction of target string pairs to which the estimated ranking assigns the same order as the true one. During testing, for each source string, we sample at random 1000 other strings from the hold-out data, and count the fraction of all pairs ordered correctly according to this criterion.

A 0-order model successfully learns to rank strings according to this affinity with 99.3% accuracy, while ranking according to the unmodified Levenshtein distance yields 76.4%. Table 3 shows the 6 averaged weights with the highest magnitude from a model trained on one of the folds. In presenting weights, we follow the formatting convention *edit*:*feature*. Since the model in question is of order 0, all features in Table 3 are the “empty feature.” Note how idempotent substitutions involving the 5 highly weighted consonants are weighted significantly higher than the remaining operations.

$\langle \texttt{d}, \texttt{d} \rangle$: $f_\emptyset$	61.1
$\langle \texttt{c}, \texttt{c} \rangle$: $f_\emptyset$	60.6
$\langle \texttt{g}, \texttt{g} \rangle$: $f_\emptyset$	60.3
$\langle \texttt{b}, \texttt{b} \rangle$: $f_\emptyset$	59.1
$\langle \texttt{f}, \texttt{f} \rangle$: $f_\emptyset$	57.0
$\langle \texttt{t}, \texttt{t} \rangle$: $f_\emptyset$	18.6

Table 3: Largest weights in a consonant-preserving edit affinity in which the first five consonants are given 5 times as much weight as others.

3.3 Rhyming

While the above problem illustrates the ability of the proposed algorithm to learn latent alignment affinities, it is expressible as a order-0 model. A somewhat more interesting problem is that of modeling groups of rhyming words. This problem is an instance of what we called the “classification” scenario in Section 2.3. Because English letters have long since lost their direct correspondence to phonemes, the problem of distinguishing rhyming English words is difficult for a knowledge-lean edit model. What’s more, the importance of a letter is dependent on context; letters near the end of a word are more likely to be significant.

We derived groups of rhyming words from the CMU pronouncing dictionary (CMU, 1995), discarding any singleton groups. This yielded 21,396 words partitioned into 3,799 groups, ranging in size from 464 words (nation, location, etc.) down to 2. We then divided the words in this data set at random into three groups for cross-validation.

Training was conducted as follows. For each word in the training set, we selected at random up to 5 rhyming words and 5 non-rhyming words. These words were ranked according to affinity with the source word under the current model. Let t be the lowest scoring rhyming word, and let t' be the highest-scoring non-rhyming word.

[2] Here and in other experiments involving the edit model, the number of epochs was set arbitrarily, and *not* based on performance on a development set. Beyond the number of epochs required for convergence, we have not observed much sensitivity in test accuracy to the number of epochs.

Model	*Precision*
Levenshtein	0.126
Longest common suffix	0.130
PTEM, Order 0	0.505
PTEM, Order 3	0.790

Table 4: Micro-averaged break-even precision on the task of grouping rhyming English words.

For each word in the hold-out set, we scored and ranked all rhyming words in the same set, as well as enough non-rhyming words to total 1000. We then recorded the precision at the point in this ranking where recall and precision are most nearly equal. Our summary statistic is the micro-averaged break-even precision.

Table 4 presents the performance of the proposed model and compares it with two simple baselines. Not surprisingly, performance increases with increasing order. The simple heuristic approaches fare quite poorly by comparison, reflecting the subtlety of the problem.

3.4 Transcription

Our work was motivated by the problem of named entity transcription. Out-of-vocabulary (OOV) terms are a persistent problem in statistical machine translation. Often, such terms are the names of entities, which typically have low corpus frequencies. In translation, the appropriate handling of names is often to transcribe them, to render them idiomatically in the target language in a way that preserves, as much as possible, their phonetic structure. Even when an OOV term is not a name, transcribing it preserves information that would otherwise be discarded, leaving open the possibility that downstream applications will be able to make use of it.

The state of the art in name transcription involves some form of generative model, sometimes in combination with additional heuristics. The generative component may involve explicitly modeling phonetics. For example, Knight and Graehl (1998) employ cascaded probabilistic finite-state transducers, one of the stages modeling the orthographic-to-phonetic mapping. Subsequently, Al-Onaizan and Knight (2002) find they can boost performance by combining a phonetically-informed model with one trained only on orthographic correspondences. Huang et al. (2004), construct a probabilistic Chinese-English edit model as part of a larger alignment solution, setting edit weights in a heuristic bootstrapped procedure.

Task	*Train*	*Dev*	*Eval*	*ELen*	*FLen*
A-E	8084	1000	1000	6.5	4.9
M-E	2000	430	1557	16.3	23.0

Table 5: Characteristics of the two transcription data sets, Arabic-English (A-E) and Mandarin-English (M-E), including number of training, development, and evaluation pairs (*Train*, *Dev*, and *Eval*), and mean length in characters of English and foreign strings (*ELen* and *FLen*).

In rendering unfamiliar written Arabic words or phrases in English, it is generally impossible to achieve perfect performance, because many sounds, such as short vowels, diphthong markers, and doubled consonants, are conventionally not written in Arabic. We calculate from our experimental datasets that approximately 25% of the characters in the English output must be inferred. Thus, a character error rate of 25% can be achieved through simple transliteration.

3.4.1 Transcribing names

We experimented with a list of 10,084 personal names distributed by the Linguistic Data Consortium (LDC). Each entry in the database includes an arabic name in transliterated ASCII (SATTS method) and its English rendering. The Arabic names appear as they would in conventional written Arabic, i.e., lacking short vowels and other diacritics. We randomly segregated 1000 entries for evaluation and used the rest for training. The *A-E* row in Table 5 summarizes some of this data set's characteristics.

We trained the edit model as follows. For each training pair the indicated English rendering constitutes our true target (t), and we use the current model to generate an alternate string (t'), updating the model in the event t' yields a higher score than t. This was repeated for 10 epochs. We experimented with a model of order 3.

Under this methodology, we observed a 1-best ac-

$\langle \texttt{p}, \texttt{h} \rangle$: $f_{t,a\_}$	38.5
$\langle \texttt{p}, \texttt{t} \rangle$: $f_{t,a\_}$	30.8
$\langle \texttt{p}, \texttt{h} \rangle$: $f_{t,ya\_}$	11.8
$\langle \texttt{p}, \texttt{t} \rangle$: $f_{s,\_p<e>}$	-8.6
$\langle \texttt{p}, \texttt{h} \rangle$: $f_{t,rya\_}$	-12.1
$\langle \texttt{p}, \texttt{t} \rangle$: $f_{t,uba\_}$	-14.4

Table 6: Some of the weights governing the handling of the *tah marbouta* (ة) in an order-3 Arabic-English location name transcription model. Buckwalter encoding of Arabic characters is used here for purposes of display. The symbol “<e>” represents end of string.

curacy of 0.552. It is difficult to characterize the strength of this result relative to those reported in the literature. Al-Onaizan and Knight (2002) report a 1-best accuracy of 0.199 on a corpus of Arabic *person* names (but an accuracy of 0.634 on *English* names), using a “spelling-based” model, i.e., a model which has no access to phonetic information. However, the details of their experiment and model differ from ours in a number of respects.

It is interesting to see how a learned edit model handles ambiguous letters. Table 6 shows the weights of some of the features governing the handling of the character ة (*tah marbouta*) from experiments with Arabic place names. This character, which represents the “t” sound, typically appears at the end of words. It is generally silent, but is spoken in certain grammatical constructions. In its silent form, it is typically transcribed “ah” (or “a”); in its spoken form, it is transcribed “at”. The weights in the table reflect this ambiguity and illustrate some of the criteria by which the model chooses the appropriate transcription. For example, the negative weight on the feature $f_{s,\_p<e>}$ inhibits the production of “t” at the end of a phrase, where “h” is almost always more appropriate. Similarly, “h” is more common following “ya” in the target string (often as part of the larger suffix “iyah”). However, the preceding context “rya” is usually observed in the word “qaryat”, meaning “village” as in “the village of ...” In this grammatical usage, the tah marbouta is spoken and therefore rendered with a “t”. Consequently, the corresponding weight in the “h” interpretation is inhibitory.

The Al-Onaizan and Knight spelling model can be regarded as a statistical machine translation (SMT) system which translates source language characters to target language characters in the absence of phonetic information. For comparison with state of the art, we used the RWTH phrase-based SMT system (Zens et al., 2005) to build an Arabic-to-English transliteration system. This system frames the transcription problem as follows. We are given a sequence of source language characters s_1^m representing a name, which is to be translated into a sequence of target language characters t_1^n. Among all possible target language character sequences, we will choose the character sequence with the highest probability:

$$\hat{t}_1^{\hat{n}} = \underset{n,t_1^n}{\operatorname{argmax}} \{\Pr(t_1^n|s_1^m)\} \qquad (3)$$

The posterior probability $\Pr(t_1^n|s_1^m)$ is modeled directly using a log-linear combination of several models (Och and Ney, 2002), including a character-based phrase translation model, a character-based lexicon model, a 4-gram character sequence model, a character penalty and a phrase penalty. The first two models are used for both directions: Arabic to English and English to Arabic. We do not use any reordering model because the target character sequence is always monotone with respect to the source character sequence. More details about the baseline system can be found in (Zens et al., 2005).

We remark in passing that while the perceptron-based edit model is a general algorithm for learning sequence alignments using simple features, the above SMT approach combines several models, some of which have been the subject of research in the fields of speech recognition and machine translation for several years. Furthermore, we made an effort to optimize the performance of the SMT approach on the tasks presented here.

Table 7 compares this system with the edit model. The difference between the 1-best accuracies of the two systems is significant at the 95% level, using the bootstrap for testing. However, we can improve on both systems by combining them. We segregated 1000 training documents to form a development set, and used it to learn linear combination coefficients over our two systems, resulting in a combined system that scored 0.588 on the evaluation set—a sta-

Model	*1best*	*5best*
SMT	0.528	0.824
PTEM, Order 3	0.552	0.803
Linear combination	0.588	0.850

Table 7: 1-best and 5-best transcription accuracies. The successive improvements in 1-best accuracy are significant at the 95% confidence level.

tistically significant improvement over both systems at the 95% confidence level.

3.4.2 Ranking transcriptions

In some applications, instead of transcribing a name in one language into another, it is enough just to rank candidate transcriptions. For example, we may be in possession of comparable corpora in two languages and the means to identify named entities in each. If we can rank the likely transcriptions of a name, we may be able to align a large portion of the transliterated named entities, potentially extending the coverage of our machine translation system, which will typically have been developed using a smaller parallel corpus. This idea is at the heart of several recent attempts to improve the handling of named entities in machine translation (Huang et al., 2004; Lee and Chang, 2003). A core component of all such approaches is a generative model similar in structure to the "spelling" model proposed by Al-Onaizan and Knight.

When ranking is the objective, we can adopt a training procedure that is much less expensive than the one used for generation. Let t be the correct transcription for a source string (s). Sample some number of strings at random (200 in the following experiments) from among the transcriptions in the training set of strings other than s. Let t' be the string having highest affinity with s, updating the model, as usual, if t' scores higher than t.

In addition to the Arabic-English corpus, we also experiment with a corpus distributed by the LDC of full English names paired with their Mandarin spelling. The *M-E* row of Table 5 summarizes characteristics of this data set. Because we are interested in an approximate comparison with similar experiments in the literature, we selected at random 2430 for training and 1557 for evaluation, which are the data sizes used by Lee and Chang (2003) for their experiments. In these experiments, the Chinese names are represented as space-separated pinyin without tonal markers.

Note that this problem is probably harder than the Arabic one, for several reasons. For one thing, the letters in a Mandarin transcription of a foreign name represent syllables, leading to a somewhat lossier rendering of foreign names in Mandarin than in Arabic. On a more practical level, this data set is noisier, occasionally containing character sequences in one string for which corresponding characters are lacking from its paired string. On the other hand, the Mandarin problem contains full names, rather than name components, which provides more context for ranking.

We trained the edit model on both data sets using both the sampling procedure outlined above and the self-generation training regime, in each case for 20 epochs, producing models of orders from 1 to 3. However, we found that the efficiency of the phrase-based SMT system described in the previous section would be limited for this task, mainly due to two reasons: the character-based phrase models due to possible unseen phrases in an evaluation corpus, and the character sequence model as all candidate transcriptions confidently belong to the target language. Therefore, to make the phrase-based SMT system robust against data sparseness for the ranking task, we also make use of the IBM Model 4 (Brown et al., 1993) in both directions. The experiments show that IBM Model 4 is a reliable model for the ranking task. For each evaluation pair, we then ranked all available evaluation transcriptions, recording where in this list the true transcription fell.

Table 8 compares the various models, showing the fraction of cases for which the true transcription was ranked highest, and its mean reciprocal rank (MRR). Both the phrase-based SMT model and the edit model perform well on this task. While the best configuration of PTEM out-performs the best SMT model, the differences are not significant at the 95% confidence level. However, compare these performance scores to those returned by the system of Lee and Chang (2003), who reported a peak MRR of 0.82 in similar experiments involving data different from ours.

The PTEM rows in the table are separated into

Model	C-E Task		A-E Task	
	ACC	MRR	ACC	MRR
SMT	0.795	0.797	0.982	0.985
SMT w/o LM	0.797	0.798	0.983	0.985
IBM_4	0.961	0.971	0.978	0.987
SMT + IBM_4	0.971	0.977	0.991	0.994
PTEM*G*, Ord. 1	0.843	0.877	0.959	0.975
PTEM*G*, Ord. 2	0.970	0.978	0.968	0.980
PTEM*G*, Ord. 3	0.975	0.982	0.971	0.983
PTEM*R*, Ord. 1	0.961	0.973	0.992	0.995
PTEM*R*, Ord. 2	0.960	0.972	0.989	0.993
PTEM*R*, Ord. 3	0.960	0.972	0.989	0.994

Table 8: Performance on two transcription ranking tasks, showing fraction of cases in which the correct transcription was ranked highest, accuracy (ACC) and mean reciprocal rank of the correct transcription (MRR).

those in which the model was trained using the same procedure as for generation (PTEM*G*), and those in which the quicker ranking-specific training regime was used (PTEM*R*). The comparison is interesting, inasmuch it does not support the conclusion that one regime is uniformly superior to the other. While generation regime yields the best performance on Arabic (using a high-order model), the ranking regime scores best on Mandarin (with a low-order model). When training a model to generate, it seems clear that more context in the form of larger n-grams is beneficial. This is particularly true for Mandarin, where an order-1 model probably does not have the capacity to generate plausible decoys.

4 Discussion

This paper is not the first to show that perceptron training can be used in the solution of problems involving transduction. Both Liang, et al (2006), and Tillmann and Zhang (2006) report on effective machine translation (MT) models involving large numbers of features with discriminatively trained weights. The training of these models is an instance of the “Generation” scenario outlined in Section 2.3. However, because machine translation is a more challenging problem than name transcription (larger vocabularies, higher levels of ambiguity, non-monotonic transduction, etc.), our general-purpose approach to generation training may be intractable for MT. Instead, much of the focus of these papers are the heuristics that are required in order to train such a model in this fashion, including feature selection using external resources (phrase tables), staged training, and generating to BLEU-maximal sequences, rather than the reference target.

Klementiev and Roth (2006) explore the use of a perceptron-based ranking model for the purpose of finding name transliterations across comparable corpora. They do not calculate an explicit alignment between strings. Instead, they decompose a string pair into a collection of features derived from character n-grams heuristically paired based on their locations in the respective strings. Thus, Klementiev and Roth, in common with the two MT approaches described above, carefully control the features used by the perceptron. In contrast to these approaches, our algorithm discovers *latent* alignments, essentially selecting those features necessary for good performance on the task at hand.

As noted in the introduction, several previous papers have proposed general, discriminatively trained sequence alignment models, as alternatives to the generative model proposed by Ristad and Yianilos. McCallum, et al. (2005), propose a conditional random field for sequence alignment, designed for the important problem of duplicate detection and information integration. Comprising two sub-models, one for matching strings and one for non-matching, the model is trained on sequence pairs explicitly labeled “match” or “non-match,” and some care is apparently needed in selecting appropriate non-matching strings. It is therefore unclear how this model would be extended to problems involving ranking or generation.

Joachims (2003) proposes SVM-align, a sequence alignment model similar in structure to that described here, but which sets weights through direct numerical optimization. Training involves exposing the model to sequence pairs, along with the correct alignment and some number of “decoy” sequences. The reliance on an explicit alignment and hand-chosen decoys yields a somewhat less flexible solution than that presented here. It is not clear whether these features of the training regime are indispensable, or whether they might be generalized to increase the approach's scope. Note that where directly maximizing the margin is feasible, it has been shown empirically to be superior to perceptron training (Altun et al., 2003).

Parker et al. (2006), propose to align sequences by gradient tree boosting. This approach has the attractive characteristic that it supports a factored representation of edits (a characteristic it shares with McCallum et al.). Although this paper does not evaluate the method on any problems from computational linguistics (the central problem is musical information retrieval), gradient tree boosting has been shown to be an effective technique for other sorts of sequence modeling drawn from computational linguistics (Dietterich et al., 2004).

5 Conclusion

Motivated by the problem of Arabic-English transcription of names, we adapted recent work in perceptron learning for sequence labeling to the problem of sequence alignment. The resulting algorithm shows clear promise not only for transcription, but also for ranking of transcriptions and structural classification. We believe this versatility will lead to other successful applications of the idea, both within computational linguistics and in other fields involving sequential learning.

Acknowledgment of support

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA/IPTO) under Contract HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

References

Y. Al-Onaizan and K. Knight. 2002. Machine transliteration of names in Arabic text. In *Proceedings of the ACL-02 workshop on computational approaches to semitic languages*.

Y. Altun, I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov support vector machines. In *Proceedings of ICML-2003*.

M. Bilenko and R. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of KDD-2003*.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2), June.

CMU. 1995. The CMU pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Version 0.6.

M. Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP-2002*.

T. Dietterich, A. Ashenfelter, and Y. Bulatov. 2004. Training conditional random fields via gradient tree boosting. In *Proceedings of ICML-2004*.

F. Huang, S. Vogel, and A. Waibel. 2004. Improving named entity translation combining phonetic and semantic similarities. In *Proceedings of HLT-NAACL 2004*.

T. Joachims. 2003. Learning to align sequences: a maximum-margin approach. Technical report, Cornell University.

A. Klementiev and D. Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of Coling/ACL 2006*.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4).

C.-J. Lee and J.S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING 2006/ACL 2006*.

A. McCallum, K. Bellare, and F. Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of UAI-2005*.

S.B. Needleman and C.D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48.

F.J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL02*, pages 295–302, Philadelphia, PA, July.

C. Parker, A. Fern, and P Tadepalli. 2006. Gradient boosting for sequence alignment. In *Proceedings of AAAI-2006*.

E.S. Ristad and P.N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20.

C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proceedings of Coling/ACL 2006*.

R. Zens, O. Bender, S. Hasan, S. Khadivi, E. Matusov, J. Xu, Y. Zhang, and H. Ney. 2005. The RWTH phrase-based statistical machine translation system. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 155–162, Pittsburgh, PA, October.

Instance Based Lexical Entailment for Ontology Population

Claudio Giuliano and **Alfio Gliozzo**
FBK-irst, Istituto per la Ricerca Scientifica e Tecnologica
I-38050, Trento, ITALY
{giuliano,gliozzo}@itc.it

Abstract

In this paper we propose an instance based method for lexical entailment and apply it to automatic ontology population from text. The approach is fully unsupervised and based on kernel methods. We demonstrate the effectiveness of our technique largely surpassing both the random and most frequent baselines and outperforming current state-of-the-art unsupervised approaches on a benchmark ontology available in the literature.

1 Introduction

Textual entailment is formally defined as a relationship between a coherent text T and a language expression, the hypothesis H. T is said to entail H, denoted by $T \rightarrow H$, if the meaning of H can be inferred from the meaning of T (Dagan et al., 2005; Dagan and Glickman., 2004). Even though this notion has been recently proposed in the computational linguistics literature, it has already attracted a great attention due to the very high generality of its settings and to the indubitable usefulness of its (potential) applications.

In this paper, we concentrate on the problem of lexical entailment, a textual entailment subtask in which the system is asked to decide whether the substitution of a particular word w with the word e in a coherent text $H_w = H^l w H^r$ generates a sentence $H_e = H^l e H^r$ such that $H_w \rightarrow H_e$, where H^l and H^r denote the left and the right context of w, respectively. For example, given the word ‘weapon’ a system may substitute it with the synonym ‘arm’, in order to identify relevant texts that denote the sought concept using the latter term. A particular case of lexical entailment is recognizing synonymy, where both $H_w \rightarrow H_e$ and $H_e \rightarrow H_w$ hold.

In the literature, slight variations of this problem are also referred to as *sense matching* (Dagan et al., 2006), *lexical reference* (Glickman et al., 2006a) and *lexical substitution* (Glickman et al., 2006b). They have been applied to a wide variety of tasks, such as semantic matching, subtitle generation and Word Sense Disambiguation (WSD). Modeling lexical entailment is also a prerequisite to approach the SemEval-2007 lexical substitution task[1], consisting of finding alternative words that can occur in given context.

In this paper, we propose to apply an approach for lexical entailment to the ontology population task. The basic idea is that if a word entails another one in a given context then the former is an instance or a subclass of the latter. This approach is intuitively appealing because lexical entailment is intrinsically an unsupervised task, therefore it does not require lexical resources, seed examples or manually annotated data sets. Unsupervised approaches are particularly suited for ontology population, whose goal is to find instances of concepts from corpora, because both corpus and the ontology sizes can scale up to millions of documents and thousands of concepts, preventing us from applying supervised learning. In addition, the top level part of the ontology (i.e., the *Tbox* in the Description Logics terminology) is very

[1] http://nlp.cs.swarthmore.edu/semeval/tasks/task10/description.shtml

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 248–256, Prague, June 2007. ©2007 Association for Computational Linguistics

often modified during the ontology engineering lifecycle, for example by introducing new concepts and restructuring the subclass_of hierarchy according to the renewed application needs required by the evolution of the application domain. It is evident that to preserve the consistency between the Tbox and the *Abox* (i.e., the set of instances and their relations) in such a dynamic ontology engineering process, supervised approaches are clearly inadequate, as small changes in the TBox will be reflected into dramatic annotation effort to keep instances in the Abox aligned.

The problem of populating a predefined ontology of concepts with novel instances implies a WSD task, as the entities in texts are ambiguous with respect to the domain ontology. For example, the entity *Washington* is both the name of a state and the name of a city. In the ontology population settings traditional WSD approaches cannot be directly applied since entities are not reported into dictionaries, making the lexical entailment alternative more viable. In particular, we model the problem of ontology population as the problem of recognizing for each mention of an entity of a particular coarse-grained type (e.g., location) the fine-grained concept (e.g., lake or mountain) that can be substituted in texts preserving the meaning. For example, in the sentence "the first man to climb the Everest without oxygen", "Everest" can be substituted with the word *mountain* preserving the meaning, while the sentence is meaningless when "Everest" is replaced with the word *lake*. Following the lexical entailment approach, the ontology population task is transformed into the problem of recognizing the term from a fine-grained set of categories (e.g., city, country, river, lake and mountain) that can be substituted in the contexts where the entity is mentioned (e.g., Everest in the example above).

The main contributions of this paper are summarized as follows. First, we propose a novel approach to lexical entailment, called Instance Based Lexical Entailment (IBLE), that allows approaching the problem as a classification task, in which a given target word (i.e., the entailing word) in a particular context is judged to entail a different word taken from a (pre-defined) set of (possible) candidate entailed words (see Section 3). Second, we exploit the IBLE approach to model the ontology population task as follows. Given a set of candidate concepts belonging to generic ontological types (e.g., people or locations), and a set of pre-recognized mentions of entities of these types in the corpus (e.g., Newton, Ontario), we assign the entity to the class whose lexicalization is more frequently entailed in the corpus. In particular, as training set to learn the fine-grained category models, we use all the occurrences of their corresponding expressions in the same corpus (e.g., we collected all occurrences in context of the word *scientist* to describe the concept `scientist`). Then, we apply the trained model to classify the pre-recognized coarse-grained entities into the fine-grained categories.

Our approach is fully unsupervised as for training it only requires occurrences of the candidate entailed words taken in their contexts. Restricted to the ontology population task, for each coarse-grained entity (e.g., location), the candidate entailed words are the terms corresponding to the fine-grained classes (e.g., lake or mountain) and the entailing words are mentions of entities (e.g., New York, Ontario) belonging to the coarse-grained class, recognized by an entity tagger.

Experiments show that our method for recognizing lexical entailment is effective for the ontology population task, reporting improvements over a state-of-the-art unsupervised technique based on contextual similarity measures (Cimiano and Völker, 2005). In addition, we also compared it to a supervised approach (Tanev and Magnini, 2006), that we regarded as an upper bound, obtaining comparable results.

2 The Ontology Population Task

Populating concepts of a predefined ontology with instances found in a corpus is a primary goal of knowledge management systems. As concepts in the ontology are generally structured into hierarchies belonging to a common ontological type (e.g., people or locations), the problem of populating ontologies can be solved hierarchically, firstly identifying instances in texts as belonging to the topmost concepts, and then assigning them to a fine-grained class. Supervised named entity recognition (NER) systems can be used for accomplishing the first step. State-of-the-art NER systems are characterized by high accuracy, but they require a large amount of training data. However, domain specific ontologies generally contains many "fine-grained" categories (e.g., particular categories of people, such as writers, scientists, and so on) and, as a consequence, supervised methods cannot be used because the annotation costs would become prohibitive.

Therefore, in the literature, the fine-grained classification task has been approached by adopting weakly supervised (Tanev and Magnini, 2006; Fleischman and Hovy, 2002) or unsupervised methods (Cimiano and Völker, 2005). Tanev and Magnini (2006) proposed a weakly supervised method that requires as training data a list of terms without context for each class under consideration. Such list can be automatically acquired from existing ontologies or other sources (i.e., database fields, web sites like Wikipedia, etc.) since the approach imposes virtually no restrictions on them. Given a generic syntactically parsed corpus containing at least each training entity twice, the algorithm learns, for each class, a feature vector describing the contexts where those entities occur. Then it compares the new (unknown) entity with the so obtained feature vectors, assigning it to the most similar class. Fleischman and Hovy (2002) approached the ontology population problem as a classification task, providing examples of instances in their context as training examples for their respective fine-grained categories.

The aforementioned approaches are clearly inadequate to recognize such fine-grained distinctions, as they would require a time consuming and costly annotation process for each particular class, that is clearly infeasible when the number of concepts in the ontology scales up. Therefore, most of the present research in ontology population is focusing on either unsupervised approaches (Cimiano and Völker, 2005) or weakly supervised approaches (Tanev and Magnini, 2006).

Unsupervised approaches are mostly based on term similarity metrics. Cimiano and Völker (2005) assign a particular entity to the fine-grained class such that the contextual similarity is maximal among the set of fine-grained subclasses of a coarse-grained category. Contextual similarity has been measured by adopting lexico-syntactic features provided by a dependency parser, as proposed in (Lin, 1998).

3 Instance Based Lexical Entailment

Dagan et al. (2006) adapted the classical supervised WSD setting to approach the sense matching problem (i.e., the binary lexical entailment problem of deciding whether a word, such as *position*, entails a different word, such as *job*, in a given context) by defining a one-class learning algorithm based on support vector machines (SVM). They train a one-class model for each entailed word (e.g., all the occurrences of the word *job* in the corpus) and, then, apply it to classify all the occurrences of the entailing words (e.g., the word *position*), providing a binary decision criterion[2]. Similarly to the WSD case, examples are represented by feature vectors describing their contexts, and then compared to the feature vectors describing the context of the target word.

In this paper, we adopt a similar strategy to approach a multi-class lexical entailment problem. The basic hypothesis is that if a word w entails e in a particular context ($H_w \rightarrow H_e$), then some of the contexts T_e^j in which e occurs in the training corpus are similar to H_w. Given a word w and an (exhaustive) set of candidate entailed words $E = \{e_1, e_2, \ldots, e_n\}$, to which we refer hereafter with the expression "substitution lexica", our goal is to select the word $e_i \in E$ that can be substituted to w in the context H_w generating a sentence H_e such that $H_w \rightarrow H_e$. In the multi-class setting, supervised learning approaches can be used. In particular, we can apply a one-versus-all learning methodology, in which each class e_i is trained from both positive (i.e., all the occurrences of e_i in the corpus) and negative examples (i.e., all the occurrences of the words in the set $\{e_j | j \neq i\}$).

Our approach is clearly a simplification of the more general lexical entailment settings, where given two generic words w and e, and a context $H = H^l w H^r$, the system is asked to decide whether w entails e or not. In fact, the latter is a binary classification problem, while the former is easier as the system is required to select "the best" option among the substitution lexicon. Of course providing such set could be problematic in many cases (e.g., it could be incomplete or simply not available for

[2] This approach resembles the pseudo-words technique proposed to evaluate WSD algorithms at the earlier stages of the WSD studies (Gale et al., 1992), when large scale sense tagged corpora were not available for training supervised algorithms.

many languages or rare words). On the other hand, such a simplification is practically effective. First of all, it allows us to provide both positive and negative examples, avoiding the use of one-class classification algorithms that in practice perform poorly (Dagan et al., 2006). Second, the large availability of manually constructed substitution lexica, such as WordNet (Fellbaum, 1998), or the use of repositories based on statistical word similarities, such as the database constructed by Lin (1998), allows us to find an adequate substitution lexicon for each target word in most of the cases.

For example, as shown in Table 1, the word *job* has different senses depending on its context, some of them entailing its direct hyponym *position* (e.g., "looking for permanent *job*"), others entailing the word *task* (e.g., "the *job* of repairing"). The problem of deciding whether a particular instance of *job* can be replaced by *position*, and not by the word *place*, can be solved by looking for the most similar contexts where either *position* or *place* occur in the training data, and then selecting the class (i.e., the entailed word) characterized by the most similar ones, in an instance based style. In the first example (see row 1), the word *job* is strongly associated to the word *position*, because the contexts of the latter in the examples 1 and 2 are similar to the context of the former, and not to the word *task*, whose contexts (4, 5 and 6) are radically different. On the other hand, the second example (see row 2) of the word *job* is similar to the occurrences 4 and 5 of the word *task*, allowing its correct substitution.

It is worthwhile to remark that, due to the ambiguity of the entailed words (e.g., *position* could also entail either *perspective* or *place*), not every occurrence of them should be taken into account, in order to avoid misleading predictions caused by the irrelevant senses. Therefore, approaches based on a more classical contextual similarity technique (Lin, 1998; Dagan, 2000), where words are described "globally" by context vectors, are doomed to fail. We will provide empirical evidence of this in the evaluation section.

Choosing an appropriate similarity function for the contexts of the words to be substituted is a primary issue. In this work, we exploited similarity functions already defined in the WSD literature, relying on the analogy between the lexical entailment and the WSD task. The state-of-the-art supervised WSD methodology, reporting the best results in most of the Senseval-3 lexical sample tasks in different languages, is based on a combination of syntagmatic and domain kernels (Gliozzo et al., 2005) in a SVM classification framework. Therefore, we adopted exactly the same strategy for our purposes.

A great advantage of this methodology is that it is totally corpus based, as it does not require neither the availability of lexical databases, nor the use of complex preprocessing steps such as parsing or anaphora resolution, allowing us to apply it on different languages and domains once large corpora are available for training. Therefore, we exploited exactly the same strategy to implement the IBLE classifier required for our purposes, defining a kernel composed by n simple kernels, each representing a different aspect to be considered when estimating contextual similarity among word occurrences. In fact, by using the closure properties of the kernel functions, it is possible to define the kernel combination schema as follows[3]:

$$K_C(x_i, x_j) = \sum_{l=1}^{n} \frac{K_l(x_i, x_j)}{\sqrt{K_l(x_j, x_j) K_l(x_i, x_i)}}, \quad (1)$$

where K_l are valid kernel functions, measuring similarity between the objects x_i and x_j from different perspectives[4].

One means to satisfy both the WSD and the lexical entailment requirements is to consider two different aspects of similarity: domain aspects, mainly related to the topic (i.e., the global context) of the texts in which the word occurs, and syntagmatic aspects, concerning the lexico-syntactic pattern in the local context. Domain aspects are captured by the domain kernel, described in Section 3.1, while syntagmatic aspects are taken into account by the syntagmatic kernel, presented in Section 3.2.

[3] Some recent works (Zhao and Grishman, 2005; Gliozzo et al., 2005) empirically demostrate the effectiveness of combining kernels in this way, showing that the combined kernel always improves the performance of the individual ones. In addition, this formulation allows evaluating the individual contribution of each information source.

[4] An exhaustive discussion about kernel methods for NLP can be found in (Shawe-Taylor and Cristianini, 2004).

Entailed	*job*		*Training*
position	... looking for permanent academic **job** in ...	1	... from entry-level through permanent **positions**.
		2	My academic **position** ...
		3	... put the lamp in the left **position** ...
task	The **job** of repairing	4	The **task** of setting up ...
		5	Repairing the engine is an hard **task**.
		6	... **task** based evaluation.

Table 1: IBLE example.

3.1 The Domain Kernel

(Magnini et al., 2002) claim that knowing the domain of the text in which the word is located is a crucial information for WSD. For example the (domain) polysemy among the `Computer_Science` and the `Medicine` senses of the word `virus` can be solved by simply considering the domain of the context in which it is located. Domain aspects are also crucial in recognizing lexical entailment. For example, the term `virus` entails `software_agent` in the `Computer_Science` domain (e.g., "The laptop has been infected by a *virus*"), while it entails *bacterium* when located in the `Medicine` domain (e.g., "HIV is a *virus*"). As argued in (Magnini et al., 2002), domain aspects can be considered by analyzing the lexicon in a large context of the word to be disambiguated, regardless of the actual word order. We refer to (Gliozzo et al., 2005) for a detailed description of the domain kernel. The simplest methodology to estimate the domain similarity among two texts is to represent them by means of vectors in the Vector Space Model (VSM), and to exploit the cosine similarity. The VSM is a k-dimensional space $\mathbb{R}^k$, in which the text t_j is represented by means of the vector $\vec{t_j}$ such that the i^{th} component of $\vec{t_j}$ is the term frequency of the term w_i in it. The similarity between two texts in the VSM is estimated by computing the cosine between them, providing the kernel function K_{VSM} that can be used as a basic tool to estimate domain similarity between texts[5].

3.2 The Syntagmatic Kernel

Syntagmatic aspects are probably the most important evidence for recognizing lexical entailment. In general, the strategy adopted to model syntagmatic relations in WSD is to provide bigrams and trigrams of collocated words as features to describe local contexts (Yarowsky, 1994). The main drawback of this approach is that non contiguous or shifted collocations cannot be identified, decreasing the generalization power of the learning algorithm. For example, suppose that the word *job* has to be disambiguated into the sentence "…permanent academic *job* in…", and that the occurrence "We offer permanent *positions*…" is provided for training. A traditional feature mapping would extract the context words w_{-1}:`academic`, w_{-2}:`permanent` to represent the former, and w_{-1}:`permanent`, w_{-2}:`offer` to index the latter. Evidently such features will not match, leading the algorithm to a misclassification.

The syntagmatic kernel, proposed by Gliozzo et al. (2005), is an attempt to solve this problem. It is based on a gap-weighted subsequences kernel (Shawe-Taylor and Cristianini, 2004). In the spirit of kernel methods, this kernel is able to compare sequences directly in the input space, avoiding any explicit feature mapping. To perform this operation, it counts how many times a (non-contiguous) subsequence of symbols u of length n occurs in the input string s, and penalizes non-contiguous occurrences according to the number of the contained gaps. To define our syntagmatic kernel, we adapted the generic definition of the sequence kernels to the problem of recognizing collocations in local word contexts. We refer to (Giuliano et al., 2006) for a detailed description of the syntagmatic kernel.

[5] In (Gliozzo et al., 2005), in addition to the standard VSM, a domain kernel, exploiting external information acquired from unlabeled data, has been also used to reduce the amount of (labeled) training data. Here, given that our approach is fully unsupervised, i.e., we can obtain as many examples as we need, we do not use the domain kernel.

4 Lexical Entailment for Ontology Population

In this section, we apply the IBLE technique, described in Section 3, to recognize lexical entailment for ontology population. To this aim, we cast ontology population as a lexical entailment task, where the fine-grained categories are the candidate entailed words, and the named entities to be subcategorized are the entailing words. Below, we present the main steps of our algorithm in details.

Step 1 By using a state-of-the-art supervised NER system, we recognize the named entities belonging to a set of coarse-grained categories (e.g., location and people) of interest for the domain.

Step 2 For all fine-grained categories belonging to the same coarse-grained type, we extract from a domain corpus all the occurrences of their lexicalizations in context (e.g., for the category `actor`, we extract all contexts where the term *actor* occurs), and use them as input to train the IBLE classifier. In this way, we obtain a multi-class classifier for each ontological type. Then, we classify all the occurrences of the named entities recognized in the first step. The output of this process is a list of tagged named entities; where the elements of the list could have been classified into different fine-grained categories even though they refer to the same phrase (e.g., the occurrences of the entity "Jack London" could have been classified both as `writer` and `actor`, depending on the contexts where they occur).

Step 3 A distinct category is finally assigned to the entities referring to the same phrase in the list. This is done on the basis of the tags that have been assigned to all its occurrences during the previous step. To this purpose, we implemented a voting mechanism. The basic idea is that an entity belongs to a specific category if its occurrences entail a particular superclass "more often than expected by chance", where the expectation is modeled on the basis of the overall distribution of fine-grained category labels, assigned during the second step, in the corpus. This intuition is formalized by applying a statistical reliability measure, that depends on the distribution of positive assignments for each class, defined by the following formula:

$$R(e, c) = \frac{P(c|e) - \mu_c}{\sigma_c}, \tag{2}$$

where $P(c|e)$ is estimated by the relative frequency of the fine-grained class c among the different occurrences of the entity e, μ_c and σ_c measure the mean and the standard deviation of the distribution $P(c|E)$, and E is an (unlabeled) training set of instances of the coarse-grained type classified by the IBLE algorithm. Finally, each entity is assigned to the category c^* such that

$$c^* = \underset{c}{argmax}\; R(e, c). \tag{3}$$

5 Evaluation

Evaluating a lexical entailment algorithm in itself is rather complex. Therefore, we performed a task driven evaluation of our system, measuring its usefulness in an ontology population task, for which evaluation benchmarks are available, allowing us to compare our technique to existing state-of-the-art approaches.

As introduced in Section 4, the ontology population task can be modeled as a lexical entailment problem, in which the fine-grained classes are the entailed words and the named entities belonging to the coarse-grained ontological type are the entailing words.

In the following, we first introduce the experimental settings (Section 5.1). Then we evaluate our technique by comparing it to state-of-the-art unsupervised approaches for ontology population (Section 5.2).

5.1 Experimental Settings

For all experiments, we adopted the evaluation benchmark proposed in (Tanev and Magnini, 2006). It considers two high-level named entity categories both having five fine-grained sub-classes (i.e., `mountain`, `lake`, `river`, `city`, and `country` as subtypes of LOCATION; `statesman`, `writer`, `athlete`, `actor`, and `inventor` are subtypes of PERSON). The authors used WordNet and Wikipedia as primary data sources for populating the evaluation ontology. In total, the ontology is populated with 280 instances which were not ambiguous (with respect to the ontology) and appeared at least twice in the English CLEF corpus[6]. Even the evaluation task is rather small and can be perceived as an artificial experimental setting, it is the best available benchmark we can use to compare our system to existing approaches in the literature, as we are not aware of other available resources.

To perform NER we used CRFs (Lafferty et al., 2001). We trained a first-order CRF on the MUC data set to annotate locations and people. In our experiments, we used the implementation provided in MALLET (McCallum, 2002). We used a standard feature set inspired by the literature on text chunking and NER (Tjong Kim Sang and Buchholz, 2000; Tjong Kim Sang and De Meulder, 2003; Tjong Kim Sang, 2002) to train a first-order CRFs. Each instance is represented by encoding all the following families of features, all time-shifted by -2,-1,0,1,2: (a) the word itself, (b) the PoS tag of the token, (c) orthographic predicates, such as *capitalization*, *upper-case*, *numeric*, *single character*, and *punctuation*, (d) gazetteers of locations, people names and organizations, (e) character-n-gram predicates for 2 $\leqslant n \leqslant$ 3.

As an (unsupervised) training set for the fine-grained categories, we exploited all occurrences in context of their corresponding terms we found in the CLEF corpus (e.g., for the category `actor` we used all the occurrences of the term *actor*). We did not use any prior estimation of the class frequency, adopting a pure unsupervised approach. Table 2 lists the fine-grained concepts and the number of the training examples found for each of them in the CLEF corpus.

As a reference for a comparison of the outcomes of this study, we used the results presented in (Tanev and Magnini, 2006) for the Class-Word and Class-Example approaches. The Class-Word approach exploits a similarity metric between terms and concepts based on the comparison of the contexts where they appear. Details of this technique can be found in (Cimiano and Völker, 2005). Tanev and Magnini (2006) proposed a variant of the Class-Word algorithm, called Class-Example, that relies on syntactic features extracted from corpus and uses as an additional input a set of training examples for each class. Overall, it required $1,194$ examples to accomplish this task.

All experiments were performed using the SVM package *LIBSVM*[7] customized to embed our own kernel. In all the experiments, we used the default parameter setting.

location		**person**	
mountain	1681	statesman	119
lake	730	writer	3436
river	1411	athlete	642
city	35000	actor	2356
country	15037	inventor	105

Table 2: Number of training examples for each class.

5.2 Results

Table 4 shows our results compared with two baselines (i.e., random and most frequent, estimated from the test data) and the two alternative approaches for ontology population described in the previous section. Our system outperforms both baselines and largely surpasses the Class-Word unsupervised method.

It is worthwhile to remark here that, being the IBLE algorithm fully unsupervised, improving the most frequent baseline is an excellent result, rarely achieved in the literature on unsupervised methods for WSD (McCarthy et al., 2004). In addition, our system is also competitive when compared to supervised approaches, being it only 5 points lower than the Class-Example method, while it does not require seed examples and syntactic parsing. This characteristic makes our system flexible and adaptable to different languages and domains.

System	**Micro F1**	**Macro F1**
RND Baseline	0.20	0.20
Class-Word	0.42	0.33
MF baseline	0.52	NA
IBLE	0.57	0.47
Class-Example	0.62	0.68

Table 3: Comparison of different ontology population techniques.

[6] `http://www.clef-campaign.org`

[7] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Finally, we performed a disaggregated evaluation of our system, assessing the performance for different ontological types and different concepts. Results show that our method performs better on larger fine-grained classes (i.e., `writer` and `country`), while the results on smaller categories are affected by low recall, even if the predictions provided by the system tends to be highly accurate. Taking into consideration that our system is fully unsupervised, this behavior is highly desirable because it implies that it is somehow able to identify the predominant class. In addition the high precision on the smaller classes can be explained by our instance based approach.

Person	**N**	**Prec**	**Rec**	**F1**
Inventor	11	1	0.18	0.31
Statesman	20	1.0	0.05	0.10
Writer	88	0.61	0.89	0.72
Actor	25	0.57	0.68	0.62
Athlete	20	1	0.1	0.18
Micro	164	0.61	0.61	0.61
Macro	5	0.83	0.38	0.52

Table 4: Performance of the IBLE approach on people.

Location	**N**	**Prec**	**Rec**	**F1**
City	23	0.35	0.26	0.30
Country	40	0.61	0.70	0.65
River	10	0.8	0.4	0.53
Mountain	5	0.25	0.2	0.22
Lake	4	0.2	0.5	0.29
Micro	82	0.50	0.50	0.50
Macro	5	0.44	0.41	0.42

Table 5: Performance of the IBLE approach on locations.

6 Conclusions and Future Work

In this paper, we presented a novel unsupervised technique for recognizing lexical entailment in texts, namely instance based lexical entailment, and we exploited it to approach an ontology population task. The basic assumption is that if a word is entailed by another in a given context, then some of the contexts of the entailed word should be similar to that of the word to be disambiguated. Our technique is effective, as it largely surpasses both the random and most frequent baselines. In addition, it improves over the state-of-the-art for unsupervised approaches, achieving performances close to the supervised rivaling techniques requiring hundreds of examples for each class.

Ontology population is only one of the possible applications of lexical entailment. For the future, we plan to apply our instance based approach to a wide variety of tasks, e.g., lexical substitution, word sense disambiguation and information retrieval. In addition, we plan to exploit our lexical entailment as a subcomponent of a more complex system to recognize textual entailment. Finally, we are going to explore more elaborated kernel functions to recognize lexical entailment and more efficient learning strategies to apply our method to web-size corpora.

Acknowledgments

The authors would like to thank Bernardo Magnini and Hristo Tanev for providing the benchmark, Ido Dagan for useful discussions and comments regarding the connections between lexical entailment and ontology population, and Alberto Lavelli for his thorough review. Claudio Giuliano is supported by the X-Media project (`http://www.x-media-project.org`), sponsored by the European Commission as part of the Information Society Technologies (IST) program under EC grant number IST-FP6-026978. Alfio Gliozzo is supported by the FIRB-Israel research project N. RBIN045PXH.

References

Philipp Cimiano and Johanna Völker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of RANLP'05*, pages 66– 166–172, Borovets, Bulgaria.

I. Dagan and O. Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. In *Proceedings of the PASCAL Workshop on Learning Methods for Text Understanding and Mining*, Grenoble.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Ido Dagan, Oren Glickman, Alfio Gliozzo, Efrat Marmorshtein, and Carlo Strapparava. 2006. Direct word sense matching for lexical substitution. In *Proceedings ACL-2006*, pages 449–456, Sydney, Australia, July.

I. Dagan. 2000. Contextual word similarity. In Rob Dale, Hermann Moisl, and Harold Somers, editors, *Handbook of Natural Language Processing*, chapter 19, pages 459–476. Marcel Dekker Inc.

C. Fellbaum. 1998. *WordNet. An Electronic Lexical Database*. MIT Press.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of ACL-2002*, pages 1–7, Morristown, NJ, USA.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In R. Goldman et al., editor, *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 54–60.

Claudio Giuliano, Alfio Massimiliano Gliozzo, and Carlo Strapparava. 2006. Syntagmatic kernels: a word sense disambiguation case study. In *Proceedings of the EACL-2006 Workshop on Learning Structured Information in Natural Language Applications*, Trento, Italy, 5-7 April.

O. Glickman, E. Shnarch, and I. Dagan. 2006a. Lexical reference: a semantic matching subtask. In *proceedings of EMNLP 2006*.

Oren Glickman, Ido Dagan, Mikaela Keller, Samy Bengio, and Walter Daelemans. 2006b. Investigating lexical substitution scoring for subtitle generation. In *Proceedings of CoNLL-2006*.

A. Gliozzo, C. Giuliano, and C. Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of ACL-2005*, pages 403–410, Ann Arbor, Michigan, June.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-2002*, pages 282–289, Williams College, MA. Morgan Kaufmann, San Francisco, CA.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of ACL-98*, pages 768–774, Morristown, NJ, USA.

B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(4):359–373.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of ACL-2004*, Barcelona, Spain, July.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Hristo Tanev and Bernardo Magnini. 2006. Weakly supervised approaches for ontology population. In *Proceedings of EACL-2006*, Trento, Italy.

Erik Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000*, Lisbon, Portugal.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147, Edmonton, Canada.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158, Taipei, Taiwan.

D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of ACL-94*, pages 88–95, Las Cruces, New Mexico.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of ACL 2005*, Ann Arbor, Michigan, June.

Recovering Non-Local Dependencies for Chinese

Yuqing Guo
NCLT, School of Computing
Dublin City University
Dublin 9, Ireland
yguo@computing.dcu.ie

Haifeng Wang
Toshiba (China)
Research and Development Center
Beijing, 100738, China
wanghaifeng@rdc.toshiba.com.cn

Josef van Genabith
NCLT, School of Computing
Dublin City University
IBM CAS, Dublin, Ireland
josef@computing.dcu.ie

Abstract

To date, work on Non-Local Dependencies (NLDs) has focused almost exclusively on English and it is an open research question how well these approaches migrate to other languages. This paper surveys non-local dependency constructions in Chinese as represented in the Penn Chinese Treebank (CTB) and provides an approach for generating proper predicate-argument-modifier structures including NLDs from surface context-free phrase structure trees. Our approach recovers non-local dependencies at the level of Lexical-Functional Grammar f-structures, using automatically acquired subcategorisation frames and f-structure paths linking antecedents and traces in NLDs. Currently our algorithm achieves 92.2% f-score for trace insertion and 84.3% for antecedent recovery evaluating on gold-standard CTB trees, and 64.7% and 54.7%, respectively, on CTB-trained state-of-the-art parser output trees.

1 Introduction

A substantial number of linguistic phenomena such as topicalisation, relativisation, coordination and raising & control constructions, permit a constituent in one position to bear the grammatical role associated with another position. These relationships are referred to Non-Local Dependencies (NLDs), where the surface location of the constituent is called “antecedent”, and the site where the antecedent should be interpreted semantically is called “trace”. Capturing non-local dependencies is crucial to the accurate and complete determination of semantic interpretation in the form of predicate-argument-modifier structures or deep dependencies.

However, with few exceptions (Model 3 of Collins, 1999; Schmid, 2006), output trees produced by state-of-the-art broad coverage statistical parsers (Charniak, 2000; Bikel, 2004) are only surface context-free phrase structure trees (CFG-trees) without empty categories and coindexation to represent displaced constituents. Because of the importance of non-local dependencies in the proper determination of predicate-argument structures, recent years have witnessed a considerable amount of research on reconstructing such hidden relationships in CFG-trees. Three strategies have been proposed: (i) post-processing parser output with pattern matchers (Johnson, 2002), linguistic principles (Campbell, 2004) or machine learning methods (Higgins, 2003; Levy and Manning, 2004; Gabbard et al., 2006) to recover empty nodes and identify their antecedents;[1] (ii) integrating non-local dependency recovery into the parser by enriching a simple PCFG model with GPSG-style gap features (Collins, 1999; Schmid, 2006); (iii) pre-processing the input sentence with a finite-state trace tagger which detects empty nodes before parsing, and identify the antecedents on the parser output with the gap information (Dienes and Dubey, 2003a; Dienes and Dubey, 2003b).

In addition to CFG-oriented approaches, a number of richer treebank-based grammar acquisition and parsing methods based on HPSG (Miyao et al., 2003), CCG (Clark and Hockenmaier, 2002), LFG (Riezler et al., 2002; Cahill et al., 2004) and Dependency Grammar (Nivre and Nilsson, 2005) incorporate non-local dependencies into their deep syntactic or semantic representations.

A common characteristic of all these approaches

[1] (Jijkoun, 2003; Jijkoun and Rijke, 2004) also describe post-processing methods to recover NLDs, which are applied to syntactic dependency structures converted from CFG-trees.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 257–266, Prague, June 2007.

is that, to date, the research has focused almost entirely on English,[2] despite the disparity in type and frequency of non-local dependencies for various languages. In this paper, we address recovering non-local dependencies for Chinese, a language drastically different from English and whose special features such as lack of morphological inflection make NLD recovery more challenging. Inspired by (Cahill et al., 2004)'s methodology which was originally designed for English and Penn-II treebank, our approach to Chinese non-local dependency recovery is based on Lexical-Functional Grammar (LFG), a formalism that involves both phrase structure trees and predicate-argument structures. NLDs are recovered in LFG f-structures using automatically acquired subcategorisation frames and finite approximations of functional uncertainty equations describing NLD paths at the level of f-structures.

The paper is structured as follows: in Section 2 we outline the distinguishing features of Chinese non-local dependencies compared to English. In Section 3 we review (Cahill et al., 2004)'s method for recovering English NLDs in treebank-based LFG approximations. In Section 4, we describe how we modify and substantially extend the previous method to recover all types of NLDs for Chinese data. We present experiments and provide a dependency-based evaluation in Section 5. Finally we conclude and summarise future work.

2 Non-Local Dependencies in Chinese

In the Penn Chinese Treebank (CTB) (Xue et al., 2002) non-local dependencies are represented in terms of empty categories (ECs) and (for some of them) coindexation with antecedents, as exemplified in Figure 1. Following previous work for English and the CTB annotation scheme, we use “non-local dependencies” as a cover term for all missing or dislocated elements represented in the CTB as an empty category (with or without coindexation/antecedent), and our use of the term remains agnostic about fine-grained distinctions between non-local dependencies drawn in the theoretical linguistics literature.

In order to give an overview on the characteristics of Chinese non-local dependencies, we extracted all empty categories together with coindexed antecedents from the Penn Chinese Treebank version 5.1 (CTB5.1). Table 1 gives a breakdown of the most frequent types of empty categories and their antecedents, which account for 43,791 of the total 43,954 (99.6%) ECs in CTB5.1.[3]

(1) 不 愿意 发掘 培植 有 潜力 的 新 作家
not want look-for train have potential DE new writer
'(People) don't want to look for and train new writers who have potential.'

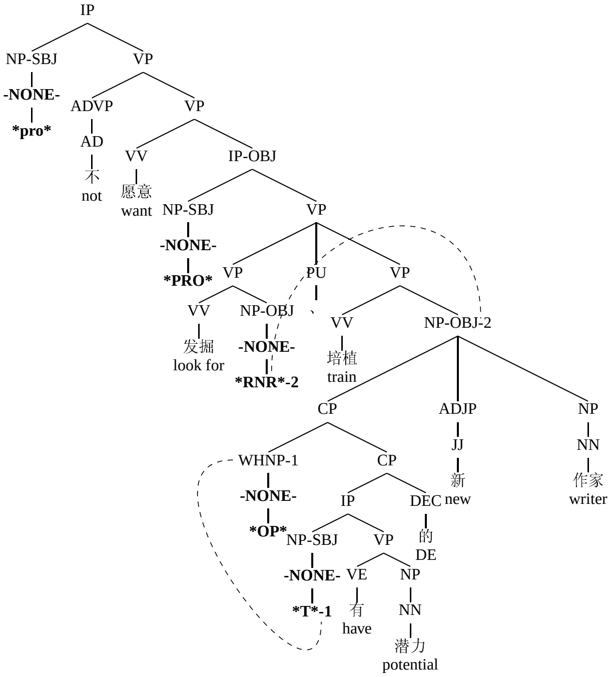


Figure 1: Example of non-local annotations in CTB, including dropped subject (*pro*), control subject (*PRO*), relative clause (*T*), and coordination (*RNR*).

According to their different linguistics properties, we divide the empty nodes listed in Table 1 into three major types: null relative pronouns, locally mediated dependencies, and long-distance dependencies.

Null Relative Pronouns (lines 2, 7) themselves are local dependencies, and thus are not coindexed with an antecedent. But they mediate non-local dependencies by functioning as antecedents for the dis-

[2] (Levy and Manning, 2004) is the only approach we are aware of that has been applied to both English and German.

[3] An extensive description of the types of empty categories and the use of coindexation in CTB can be found in Section VI of the bracketing guidelines.

	Antecedent	POS	Label	Count	Description
1	WHNP	NP	*T*	11670	WH trace (e.g. *OP*中国/China发射/launch*T*的/DE卫星/satellite)
2		WHNP	*OP*	11621	Empty relative pronouns (e.g. *OP*中国/China发射/launch的/DE卫星/satellite)
3		NP	*PRO*	10946	Control constructions (e.g. 这里/here不/not许/allow*PRO*抽烟/smoke)
4		NP	*pro*	7481	Pro-drop situations (e.g. *pro*不/not曾/ever遇到/encounter的/DE问题/problem)
5	IP	IP	*T*	575	Topicalisation (e.g. 我们/we能/can赢/win，他/he说/say*T*)
6	WHPP	PP	*T*	337	WH trace (e.g. *OP*人口/population*T*密集/dense地区/area)
7		WHPP	*OP*	337	Empty relative pronouns (e.g. *OP*人口/population密集/dense地区/area)
8	NP	NP	*	291	Raising & passive constructions (e.g. 我们/we被/BEI排除/exclude*在外/outside)
9	NP	NP	*RNR*	258	Coordinations (e.g. 鼓励/encourage*RNR*和/and支持/support投资/investment)
10	CLP	CLP	*RNR*	182	Coordinations (e.g. 五/five*RNR*至/to十/ten亿/hundred million元/Yuan)
11	NP	NP	*T*	93	Topicalisation (e.g. 薪水/salary都/all用/use*T*来/for享乐/pleasure)

Table 1: The distribution of the most frequent types of empty categories and their antecedents in CTB5.1. The types with frequency less than 30 are ignored.

located constituent inside a relative clause.[4]

Locally Mediated Dependencies are non-local as they are projected through a third lexical item (such as a control or raising verb) which involves a dependency between two adjacent levels and they are therefore bounded. This type encompasses: (line 8) raising constructions, and short-bei constructions (passivisation); (line 3) control constructions, which includes two different types: a generic *PRO* with an arbitrary reading (approximately equals to unexpressed subjects of *to*-infinitive and gerund verbs in English); and a *PRO* with definite reference (subject or object control).[5]

Long-Distance Dependencies (LDDs) differ from locally mediated dependencies, in that the path linking the antecedent and trace might be unbounded (also called unbounded, long-range dependencies). LDDs include the following phenomena:

Wh-traces in relative clauses, where an argument (line 1) or adjunct (line 6) “moves” and is coindexed with the “extraction” site.

Topicalisation (lines 5, 11) is one of the typical LDDs in English, whereas in Chinese not all topics involve displacement, for instance (2).

(2) 北京 秋天 最 美
Beijing autumn most beautiful
‘Autumn is the most beautiful in Beijing.’

Coordination is divided into two groups: right node raising of an NP phrase which is an argument shared by the coordinate predicates (line 9); and the coordination of quantifier phrases (line 10) and verbal phrases (3), in which the antecedent and trace are both predicates and possibly take their own arguments or adjuncts.

(3) 我 和 他 分别 去 公司 和 *RNR* 医院
I and he respectively go to company and *RNR* hospital
‘I went to the company and he went to the hospital respectively.’

Pro-drop situations (line 4) are prominent in Chinese because subject and object are only semantically but not syntactically required. Nevertheless we also treat pro-drop as a long-distance dependency as in principle the dropped subjects can be determined from the general (often inter-sentential) context.

Table 2 gives a quantitative comparison of NLDs between Chinese data in CTB5.1 and English in Penn-II. The data reveals that: first, NLDs in Chinese are much more frequent than in English (by nearly 1.5 times); and moreover 69% are not explicitly linked to an antecedent, compared to 43% for English, due to the high prevalence of pro-drop in Chinese.

	# of sent	# of EC	# of EC/sent	# non-coindex	% non-coindex
Chinese	18,804	43,954	2.34	30,429	69.23
English	49,207	79,245	1.61	34,455	43.48

Table 2: Comparison of NLDs between Chinese data in CTB5.1 and English in Penn-II .

[4] Null relative pronouns used in the CTB annotation are to distinguish relative clauses in which an argument or adjunct of the embedded verb is “missing” from complement (appositive) clauses which do not involve non-local dependencies.

[5] However in this case the CTB annotation doesn’t coindex the locus (trace) with its controller (antecedent).

(4) 钱 我们 用 来 享乐
money we use to please
'Money, we use for pleasure.'

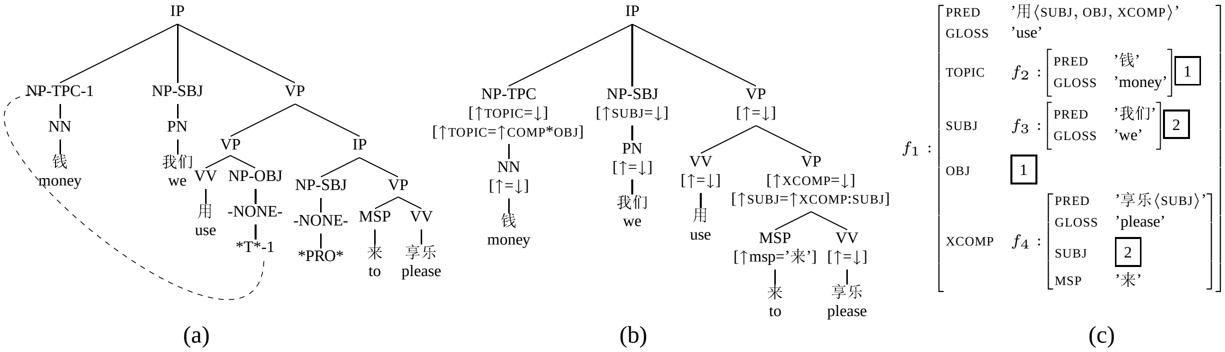


Figure 2: (a) the CTB tree; (b) LFG c-structure with functional equations; (c) corresponding f-structure. (↑) in the functional annotation refers to the f-structure associated with the mother node and (↓) to that of the local node.

3 NLD Recovery in LFG Approximations

3.1 Lexical Functional Grammar

Lexical Functional Grammar (Kaplan and Bresnan, 1982) is a constraint-based grammar formalism which minimally involves two levels of syntactic representation: c(onstituent)-structure and f(unctional)-structure. C-structure takes the form of CFG-trees and captures surface grammatical configurations. F-structure encodes more abstract grammatical functions (GFs) such as SUBJ(ect), OBJ(ect), COMP(lement), ADJ(unct) and TOPIC etc., in the form of Attribute Value Matrices which approximate to basic predicate-argument-adjunct structures or dependency relations. C-structures are related to f-structures by functional annotations (cf. Figure 2 (b) & (c)).

In LFG, non-local dependencies are captured at f-structure level in terms of reentrancies, indicated $\boxed{1}$ for the topicalisation and $\boxed{2}$ for the control construction in Figure 2(c) obviating the need for traces and coindexation in the c-structure (Figure 2(b)), unlike in CTB trees (Figure 2(a)). LFG uses functional uncertainty (FU) equations (regular expressions) to specify paths in f-structures between the trace and its antecedent. To account for the reentrancy $\boxed{1}$ in the f-structure, a FU equation of the form ↑TOPIC=↑COMP*OBJ is required (as the length of the dependency might be unbounded). The equation states that the value of the TOPIC attribute is token identical with the value of the final OBJ argument along a path through the immediately enclosing f-structure along zero or more COMP attributes.

In addition to FU equations, subcategorisation information is also a significant ingredient in LFG's account of non-local dependencies. Subcategorisation frames (subcat frames) specify the governable grammatical functions (i.e. arguments) required by a particular predicate. In Figure 2(c) each predicate in the f-structure is followed by its subcat frame.

3.2 F-Structure Based NLD Recovery

(Cahill et al., 2004) presented a NLD recovery algorithm operating at LFG f-structure for treebank-based LFG approximations. The method automatically converts Penn-II treebank trees with traces and coindexation into proper f-structures where traces and coindexation in treebank trees (Figure 2(a)) are represented as corresponding reentrances in f-structures (Figure 2(c)), and from the f-structures automatically extracts subcat frames by collecting all arguments of the local predicate at each level of the f-structures, and further acquires finite approximations of FU equations by extracting paths linking the reentracies occurring in the f-structures.

(Cahill et al., 2004)'s approach for English resolves three LDD types in parser output trees without traces and coindexation (Figure 2(b)), i.e. topicalisation (TOPIC), wh-movement in relative clauses (TOPIC_REL) and interrogatives (FOCUS). Given a set of subcat frames s for lemma w with probabilities $P(s|w)$, a set of paths p linking reentrancies conditioned on the triggering antecedent a (TOPIC, TOPIC_REL or FOCUS) with probabilities $P(p|a)$, the core algorithm recursively traverses an f-structure f to:

- find a TOPIC|TOPIC_REL|FOCUS:g pair;
- traverse f along path p to the sub-f-structure h;
- retrieve the local PRED:w at h, and insert g to h iff
  - * all GFs specified in the subcat frame s except g are present at h (completeness condition)
  - * no other governable GFs present at h are specified in s (coherence condition)
- rank resolution candidates according to the product of subcat frame and NLD path probabilities (Eq. 1).

$$P(s|w) \times P(p|a) \tag{1}$$

4 NLD Recovery Algorithm for Chinese

4.1 Automatic F-Structure Generation

Our NLD recovery is done at the level of LFG f-structures. Inspired by (Cahill et al., 2004; Burke et al., 2004), we have implemented an f-structure annotation algorithm to automatically obtain f-structures from CFG-trees in the CTB5.1. The f-structure annotation algorithm, described below, is applied both to the original CTB trees providing functional tags, traces and coindexation to generate the training corpus, and to the parser output trees without traces and coindexation to provide the f-structure input for NLD recovery.

1. The CFG-trees are head-lexicalised by head-finding rules similar to (Collins, 1999), adapted to CTB.
2. Each local subtree of depth one is partitioned by the head into left and right context. Left-right context rules exploiting configurational, categorial and CTB functional tag information are used to assign each left and right constituent with appropriate functional equations.
3. Empty nodes and coindexation in the CTB trees are automatically captured into corresponding reentrances at f-structure via functional equations.
4. All the functional equations are collected and then passed to a constraint solver to generate f-structures.

4.2 Adaptation to Chinese

(Cahill et al., 2004)'s algorithm (Section 3.2) only resolves certain NLDs with known types of antecedents (TOPIC, TOPIC_REL and FOCUS) at f-structures. However, as illustrated in Section 2, except for relative clauses, the antecedents in Chinese NLDs do not systematically correspond to types of grammatical function. Furthermore nearly 70% of all empty categories are not coindexed with an antecedent. In order to resolve all Chinese NLDs represented in the CTB, we modify and substantially extend the (Cahill et al., 2004) (henceforth C04 for short) algorithm as follows:

Given the set of subcat frames s for the word w, and a set of paths p for the trace t, the algorithm traverses the f-structure f to:

- predict a dislocated argument t at a sub-f-structure h by comparing the local PRED:w to w's subcat frames s
- t can be inserted at h if h together with t is complete and coherent relative to subcat frame s
- traverse f starting from t along the path p
- link t to it's antecedent a if p's ending GF a exists in a sub-f-structure within f; or leave t without an antecedent if an empty path for t exists

In the modified algorithm, we condition the probability of NLD path p (including the empty path without an antecedent) on the GF associated of the trace t rather than the antecedent a as in C04. The path probability $P(p|t)$ is estimated as:

$$P(p|t) = \frac{count(p,t)}{\sum_{i=1}^{n} count(p_i,t)} \tag{2}$$

In contrast even to English, Chinese has very little morphological information. As a result, every word in Chinese has a unique form regardless of its syntactic distribution. For this reason we use more syntactic features w_feats in addition to word form to discriminate between appropriate subcat frames s. For a given word w, w_feats include:

- w_pos: the part-of-speech of w
- w_gf: the grammatical function of w

$P(s|w, w\_feats)$ replaces C04's $P(s|w)$ as lexical subcat frame probability and is estimated as:

$$P(s|w, w\_feats) = \frac{count(s, w, w\_feats)}{\sum_{i=1}^{n} count(s_i, w, w\_feats)} \quad (3)$$

As more conditioning features may cause sever sparse-data problems, in order to increase the coverage of the automatically acquired subcat frames, the subcat frame frequencies $count(s, w, w\_feats)$ are smoothed by backing off to w's part-of-speech w_pos according to Eq. (4). $P(s|w\_pos)$ is estimated according to Eq. (5) and weighted by a parameter Θ. The lexical subcat frame probabilities are estimated from the smoothed frequencies as shown in Eq. (6).

$$\begin{aligned} count_{bk}(s, w, w\_feats) &= count(s, w, w\_feats) \quad (4) \\ &\quad + \Theta P(s|w\_pos) \\ P(s|w\_pos) &= \frac{count(s, w\_pos, w\_gf)}{\sum_{i=1}^{n} count(s_i, w\_pos, w\_gf)} \quad (5) \\ P_{bk}(s|w, w\_feats) &= \frac{count_{bk}(s, w, w\_feats)}{\sum_{i=1}^{n} count_{bk}(s_i, w, w\_feats)} \quad (6) \end{aligned}$$

Finally, NLD resolutions are ranked according to:

$$P_{bk}(s|w, w\_feats) \times \prod_{j=1}^{m} P(p|t_j) \quad (7)$$

As, apart from the maximum number of arguments in a subcat frame, there is no a priori limit on the number of dislocated arguments in a local f-structure, we rank resolutions with the product of the path probabilities of each (of m) missing argument(s).

4.3 A Hybrid Fine-Grained Strategy

As described in Section 2, there are three types of NLDs in the CTB, and their different linguistic properties may require fine-grained recovery strategies. Furthermore, as the NLD recovery method described in Section 4.2 is triggered by "missing" subcategorisable grammatical functions, a few cases of NLDs in which the trace is not an argument in the f-structure, e.g. an ADJUNCT or TOPIC in relative clauses or an null PRED in verbal coordination, can not be recovered by the algorithm. Table 3 shows the types of NLD that can be recovered by C04 and by the algorithm presented in Section 4.2. Table 3 shows that a hybrid methodology is required to resolve all types of NLDs in the CTB. The hybrid method involves four strategies:

- Applying a few simple heuristic rules to insert the empty PRED for coordinations and null relative pronouns for relative constructions. The former is done by comparing the part-of-speech of the local predicates and their arguments in each coordinate; and the latter is triggered by GF ADJUNCT_REL in our system.
- Inserting an empty node with GF SUBJ for short-bei construction, control and raising constructions, and relate it to the upper-level SUBJ or OBJ accordingly.
- Exploiting the C04 algorithm to resolve the wh-trace in relativisation, including ungovernable GFs TOPIC and ADJUNCT.
- Using our modified algorithm (Section 4.2) to resolve the remaining types, viz. long-distance dependencies in Chinese.

	Antecedent			Trace	
	Topic_Rel	Other	Null	Argument	Adjunct
C04	√			√	√
Ours	√	√	√	√	

Table 3: Comparison of the ability of NLD recovery for Chinese between C04 and our algorithm

5 Experiments and Evaluation

For all our experiments, we used the first 760 articles (chtb_001.fid to chtb_931.fid, 10,384 sentences) of CTB5.1, from which 75 double-annotated files (chtb_001.fid to chtb_043.fid and chtb_900.fid to chtb_931.fid, 1,046 sentences) were used as test data,[6] 75 files (chtb_306.fid to chtb_325.fid and chtb_400.fid to chtb_454.fid, 1,082 sentences) were held out as development data, while the other 610 files (8,256 sentences) were used as training data. Experiments were carried out on two different kinds of input: first on CTB gold standard trees stripped of all empty nodes and coindexation information; and

[6] The complete list of double-annotated files can be found in the documentation of CTB5.1.

second, on the output trees of Bikel's parser (Bikel, 2004).

The evaluation metric adopted by most previous work used the label and string position of the trace and its antecedent (Johnson, 2002). As pointed out by (Campbell, 2004), this metric is insensitive to the correct attachment of the EC into the parse tree, and more importantly it is not clear whether it adequately measures performance in predicate-argument structure recovery. Therefore, we use a predicate-argument based evaluation method instead. The NLD recovery is represented as a triple in the form of REL(PRED : loc, GF : loc), where REL is the relation between the dislocated GF and the PRED. In the evaluation for insertion of traces, the GF is represented by the empty category, and in the evaluation for antecedent recovery, the GF is realised by the predicate of the antecedent, e.g. OBJ(用/use:3, 钱/money:1) in Figure 2(c). The antecedent and PRED are both numbered with their string position in the input sentence. Precision, recall and f-score are calculated for the evaluation.

5.1 CTB-Based F-Structure and NLD Resources Acquisition

5.1.1 Automatically Acquired F-Structures

As described in Section 4.1, we automatically generate LFG f-structures from the CTB trees to obtain the training data and generate f-structures from the parser output trees, on which the NLDs will be recovered. To evaluate the performance of the automatic f-structure annotation algorithm, we randomly selected 200 sentences from the test set and manually annotated the f-structures to generate a gold standard. The evaluation metric is the same as for NLD recovery in terms of predicate-argument relations. Table 4 reports the results against the 200-sentence gold standard given the original CTB trees and trees output by Bikel's parser.

Dependencies	Precision	Recall	F-Score
CTB Trees	95.60	95.82	95.71
Parser Output	74.37	73.15	73.75

Table 4: Evaluation of f-structure annotation

5.1.2 Acquiring Subcat Frames and NLD Paths

From the automatically generated f-structure training data, we extract 144,119 different lexical subcat frames and 178 paths linking traces and antecedents for NLD recovery. Tables 5 & 6 show some examples of the automatically extracted subcat frames and NLD paths respectively.

Word:POS-GF(Subcat Frames)	**Prob.**
创立:VV-adj_rel([subj,obj])	0.7655
创立:VV-adj_rel([subj])	0.1537
创立:VV-adj_rel([subj,xcomp])	0.0337
......	...
创立:VV-coord([subj,obj])	0.7915
创立:VV-coord([subj])	0.0975
......	...
创立:VV-top([subj,obj])	0.5247
创立:VV-top([subj,comp])	0.2077
......	...

Table 5: Examples of subcat frames

Trace (Path)	**Prob.**
adjunct(up-adjunct:down-topic_rel)	0.9018
adjunct(up-adjunct:up-coord:down-topic_rel)	0.0192
adjunct(NULL)	0.0128
......	...
obj(up-obj:down-topic_rel)	0.7915
obj(up-obj:up-coord:down-coord:down-obj)	0.1108
......	...
subj(NULL)	0.3903
subj(up-subj:down-topic_rel)	0.2092
......	...

Table 6: Examples of NLD paths

5.2 The Basic Model

The basic algorithm described in Section 4.2 can be used to indiscriminately resolve almost all NLD types for Chinese including locally mediated dependencies with few exceptions (traces with modifier GFs, which accounts for about 1.5% of all NLDs in CTB5.1). Table 7 shows the results of the basic algorithm for trace insertion and antecedent recovery on both stripped CTB trees and parser output trees. For comparison, we implemented the C04 algorithm on our data and evaluated the result. Since the basic algorithm focus on argument traces, results for arguments only are given separately.

Table 7 shows that the C04 algorithm achieves a high precision but as expected a low recall due to its limitation to certain types of NLDs. By contrast, our basic algorithm scored higher recall but lower precision, which is understandable as the C04 algorithm identifies the trace given a known antecedent, whereas our algorithm tries to identify both the trace and antecedent. Compared to trace

	Insertion						Recovery					
	CTB Trees			Parser Output			CTB Trees			Parser Output		
	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
(Cahill et al., 2004)												
overall	**95.98**	57.86	72.20	**73.00**	40.28	51.91	**90.16**	54.35	67.82	**65.54**	36.16	46.61
args_only	**98.64**	42.03	58.94	**82.69**	30.54	44.60	**86.36**	36.80	51.61	**66.08**	24.40	35.64
Basic Model												
overall	92.44	91.28	**91.85**	63.87	62.15	**63.00**	63.12	62.33	62.72	42.69	41.54	42.10
args_only	89.42	92.95	**91.15**	60.89	63.45	**62.15**	47.92	49.81	48.84	31.41	32.73	32.06
Basic Model with Subject Path Constraint												
overall	92.16	**91.36**	91.76	63.72	**62.20**	62.95	75.96	**75.30**	**75.63**	50.82	**49.61**	**50.21**
args_only	89.04	**93.08**	91.02	60.69	**63.52**	62.07	66.15	**69.15**	**67.62**	42.77	**44.76**	**44.76**

Table 7: Evaluation of trace insertion and antecedent recovery for C04 algorithm, our basic algorithm and basic algorithm with the subject path constraint.

	Insertion						Recovery					
	Basic Model			Hybrid Model			Basic Model			Hybrid Model		
	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
Overall	92.16	91.36	91.76	**92.86**	**91.45**	**92.15**	75.96	75.30	75.63	**84.92**	**83.64**	**84.28**
SUBJ	92.95	97.81	95.32	94.38	97.81	96.06	66.93	70.42	68.63	81.61	84.57	83.06
OBJ	65.28	64.98	65.13	78.95	55.30	65.04	61.57	61.29	61.43	75.66	53.00	62.33
ADJUNCT	0.0	0.0	0.0	38.24	25.49	30.59	0.0	0.0	0.0	38.24	25.49	30.59
TOPIC	0.0	0.0	0.0	33.33	35.14	34.21	0.0	0.0	0.0	33.33	35.14	34.21
TOPIC_REL	99.85	99.39	99.62	99.85	99.39	99.62	99.85	99.39	99.62	99.85	99.39	99.62
COORD	90.00	100.00	94.74	90.00	100.00	94.74	90.00	100.00	94.74	90.00	100.00	94.74

Table 8: Breakdown of trace insertion and antecedent recovery results on stripped CTB trees for the hybrid model by major grammatical functions.

insertion, the general results for antecedent identification are rather poor. Examining the development data, we found that most recovery errors were due to wrongly treating missing SUBJs as a PRO (using empty NLD paths). Since the subject in Chinese has a very strong tendency to be omitted if it can be inferred from context, the empty NLD path (without any antecedent) has the greatest probability in all resolution paths conditioned on SUBJ, and prevents the SUBJ from finding a proper antecedent in certain cases. To test the effect of the empty path on SUBJ, we weighted non-empty paths for SUBJ so as to suppress the empty path. After testing on the development set, the optimal weight was found to be 1.9. The subject path constraint model shows a dramatic improvement of 12.9% and 8.1% for the overall result of antecedent recovery on CTB trees and parser output trees.

5.3 The Hybrid Fine-Grained Model

As proposed in Section 4.3, we implemented a more fine-grained strategy to capture specific linguistic properties of different NLD types in the CTB. We also combine our basic algorithm (Section 4.2) with (Cahill et al., 2004)'s algorithm in order to resolve the modifier-function traces. The two algorithms may conflict due to (i) inserting the same trace at the same site but related to different antecedents or (ii) resolving the same antecedent to different traces. We keep the traces inserted by the C04 algorithm and abandon those inserted by our algorithm in case of conflict, as the results in Section 5.2 suggest that C04 has a higher precision than ours. Table 8 reports the results of trace insertion and antecedent recovery, respectively, on stripped CTB trees, broken down by major GFs.

The fine-grained hybrid model allows us to recover NLDs with traces with modifier functions and, more importantly it is sensitive to particular linguistic properties of different NLD types. As the hybrid model separates the locally mediated dependencies from other long-distance dependencies, it increases the f-score by 8.7% for antecedent recovery compared with the basic model. Table 9 reports the results of the hybrid model on parser output trees, which shows an increase of 3.6% for antecedent recovery (compared with Table 7).

	Insertion			Recovery		
	Prec.	Rec.	F	Prec.	Rec.	F
overall	64.07	62.37	63.21	54.53	53.08	53.79

Table 9: Evaluation of hybrid model for trace insertion and antecedent recovery on parser output trees.

5.4 Better Training for Parser Output

Our experiments show that although our NLD recovery algorithm performs well on stripped CTB trees, it is sensitive to the noise in parser output trees, with a performance drop of about 30%. This is in contrast to English data, on which (Johnson, 2002) reports a drop of 7-9% moving from treebank trees to parser output trees. No doubt this is partially due to the poor performance of the parser on Chinese data. It is widely accepted that parsing Chinese is more difficult than parsing other more configurational or richer morphological languages, such as English.[7] Our NLD recovery algorithm runs on automatically generated LFG f-structures. The f-structure annotation algorithm is highly tailored to the CTB bracketing scheme (using configurational, categorial and functional tag information), and suffers considerably from errors produced by the parser. Table 4 shows that performance of the f-structure annotation decreases sharply (about 22%) for the parser output trees and this contributes to the eventual trace insertion and antecedent recovery performance drop.

Since the f-structures automatically generated from parser output trees are substantially different from those generated from the original CTB trees, our method to obtain the NLD resolution training data suffers from a serious drawback: the training data come from perfect CTB trees, whereas test data are derived from imperfect parser output trees. This constitutes a serious drawback for machine learning based approaches, such as ours: ideally, instances seen during training should be similar to unseen test data. To make training examples more similar to test instances, we reparse the training set to obtain better training data. To avoid running the parser on the training data, we carried out 10-fold-cross training, dividing the training data into 10 parts and parsing each part in turn with the parser trained on the remaining 9 parts. The reparsed training data are more similar to the test data than the original perfect CTB trees. We then converted both the reparsed training data and the original CTB trees into f-structures, and by comparing with the f-structures generated from the original CTB trees, we recovered the empty nodes and coindexation on the f-structures generated from the reparsed training data. We used parser output based f-structures to train our NLD recovery model and recovered NLDs for parser output trees from the test data. Table 10 presents the results for trace insertion and antecedent recovery on parser output trees using the improved training method, which shows a clear increase in precision and almost the same recall over the normal training (Table 9).

	Insertion			Recovery		
	Prec.	Rec.	F	Prec.	Rec.	F
overall	67.29	62.33	64.71	56.88	52.69	54.71

Table 10: Evaluation of hybrid model for trace insertion and antecedent recovery on parser output trees with better training.

6 Conclusion

We have presented an algorithm for recovering nonlocal dependencies for Chinese. Our method revises and considerably extends the approach of (Cahill et al., 2004) originally designed for English, and, to the best of our knowledge, is the first NLD recovery algorithm for Chinese. The evaluation shows that our algorithm considerably outperforms (Cahill et al., 2004)'s with respect to Chinese data.

In future work, we will refine and extend the conditioning features in our models to discriminate subcat frames and explore the possibilities to use the Chinese Propbank and Hownet to supplement our automatically acquired subcat frames. We will investigate ways of closing the gap between the performance of gold-standard and parer output trees, including improving parsing result for Chinese. We also plan to adapt other NLD recovery methods (Jijkoun and Rijke, 2004; Schmid, 2006) to Chinese and compare them with the current results.

Acknowledgements

This research is funded by Science Foundation Ireland grant 04/IN/I527.

[7] (Bikel, 2004) reports 89% f-score for English parsing of Penn-II treebank data and 79% f-score for Chinese parsing on CTB version 3.

References

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith and Andy Way. 2004. Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 320-327. Barcelona, Spain.

Daniel M. Bikel. 2004. On the Parameter Space of Generative Lexicalized Statistical Parsing Models. *Ph.D. thesis*, Department of Computer & Information Science, University of Pennsylvania. Philadelphia, PA.

Derrick Higgins. 2003. A machine-learning approach to the identification of WH gaps. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, 99-102. Budapest, Hungary.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132-139. Seattle, WA.

Helmut Schmid. 2006. Trace Prediction and Recovery With Unlexicalized PCFGs and Slash Features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 177-184. Sydney, Australia.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 99-106. Ann Arbor, Michigan.

Mark Johnson. 2002. A Simple Pattern-Matching Algorithm for Recovering Empty Nodes and Their Antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136-143. Philadelphia, PA.

Michael Burke, Olivia Lam, Rowena Chan, Aoife Cahill, Ruth O'Donovan, Adams Bodomo, Josef van Genabith and Andy Way. 2004. Treebank-Based Acquisition of a Chinese Lexical-Functional Grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*, pages 161-172, Tokyo, Japan.

Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. *Ph.D. thesis*, Department of Computer & Information Science, University of Pennsylvania. Philadelphia, PA.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a Large-Scale Annotated Chinese Corpus. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1100-1106. Taipei, Taiwan.

Péter Dienes and Amit Dubey. 2003a. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431-438. Sapporo, Japan.

Péter Dienes and Amit Dubey. 2003b. Antecedent Recovery: Experiments with a Trace Tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 33-40. Sapporo, Japan.

Richard Campbell. 2004. Using Linguistic Principles to Recover Empty Categories. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 645-652. Barcelona, Spain.

Roger Levy and Christopher D. Manning. 2004. Deep Dependencies from Context-Free Statistical Parsers: Correcting the Surface Dependency Approximation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 327-334. Barcelona, Spain.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical Functional Grammar: a Formal System for Grammatical Representation. *The Mental Representation of Grammatical Relations*, pages 173-282. MIT Press, Cambridge, MA.

Ryan Gabbard, Seth Kulick, and Mitch Marcus. 2006. Fully Parsing the Penn Treebank In *Proceedings of the Human Language Technology Conference / North American Chapter of the Association of Computational Linguistics*, pages 184-191. New York, USA.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271-278. Philadelphia, PA.

Stephen Clark and Julia Hockenmaier. 2002. Building Deep Dependency Structures with a Wide-CoverageCCG Parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 327-334. Philadelphia, PA.

Valentin Jijkoun. 2003. Finding Non-Local Dependencies: Beyond Pattern Matching. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 37-43. Sapporo, Japan.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the Output of a Parser Using Memory-Based Learning. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 311-318. Barcelona, Spain.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Probabilistic Modeling of Argument Structures Including Non-Local Dependencies. In *Proceedings of the 2003 Conference on Recent Advances in Natural Language Processing*, pages 285-291. Philadelphia, PA.

Exploiting Multi-Word Units in History-Based Probabilistic Generation

Deirdre Hogan, Conor Cafferkey, Aoife Cahill[*] and **Josef van Genabith**
National Centre for Language Technology
School of Computing, Dublin City University
Dublin 9, Ireland
dhogan,ccafferkey,josef@computing.dcu.ie

Abstract

We present a simple history-based model for sentence generation from LFG f-structures, which improves on the accuracy of previous models by breaking down PCFG independence assumptions so that more f-structure conditioning context is used in the prediction of grammar rule expansions. In addition, we present work on experiments with named entities and other multi-word units, showing a statistically significant improvement of generation accuracy. Tested on section 23 of the Penn Wall Street Journal Treebank, the techniques described in this paper improve BLEU scores from 66.52 to 68.82, and coverage from 98.18% to 99.96%.

1 Introduction

Sentence generation, or surface realisation, is the task of generating meaningful, grammatically correct and fluent text from some abstract semantic or syntactic representation of the sentence. It is an important and growing field of natural language processing with applications in areas such as transfer-based machine translation (Riezler and Maxwell, 2006) and sentence condensation (Riezler et al., 2003). While recent work on generation in restricted domains, such as (Belz, 2007), has shown promising results there remains much room for improvement particularly for broad coverage and robust generators, like those of Nakanishi et al. (2005) and Cahill and van Genabith (2006), which do not rely on handcrafted grammars and thus can easily be ported to new languages.

This paper is concerned with sentence generation from Lexical-Functional Grammar (LFG) f-structures (Kaplan, 1995). We present improvements in previous LFG-based generation models firstly by breaking down PCFG independence assumptions so that more f-structure conditioning context is included when predicting grammar rule expansions. This history-based approach has worked well in parsing (Collins, 1999; Charniak, 2000) and we show that it also improves PCFG-based generation.

We also present work on utilising named entities and other multi-word units to improve generation results for both accuracy and coverage. There has been a limited amount of exploration into the use of multi-word units in probabilistic parsing, for example in (Kaplan and King, 2003) (LFG parsing) and (Nivre and Nilsson, 2004) (dependency parsing). We are not aware of any similar work on generation. In the LFG-based generation algorithm presented by Cahill and van Genabith (2006) complex named entities (i.e. those consisting of more than one word token) and other multi-word units can be fragmented in the surface realization. We show that the identification of such units may be used as a simple measure to constrain the generation model's output.

We take the generator of (Cahill and van Genabith, 2006) as our baseline generator. When tested on f-structures for all sentences from Section 23 of the Penn Wall Street Journal (WSJ) treebank (Mar-

[*] Now at the Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Azenbergstrae 12, D-70174 Stuttgart, Germany. aoife.cahill@ims.uni-stuttgart.de

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 267–276, Prague, June 2007. ©2007 Association for Computational Linguistics

cus et al., 1993), the techniques described in this paper improve BLEU score from 66.52 to 68.82. In addition, coverage is increased from 98.18% to almost 100% (99.96%).

The remainder of the paper is structured as follows: in Section 2 we review related work on statistical sentence generation. Section 3 describes the baseline generation model and in Section 4 we show how the new history-based model improves over the baseline. In Section 5 we describe the source of the multi-word units (MWU) used in our experiments and the various techniques we employ to make use of these MWUs in the generation process. Section 6 gives experimental details and results.

2 Related Work on Statistical Generation

In (statistical) generators, sentences are generated from an abstract linguistic encoding via the application of grammar rules. These rules can be handcrafted grammar rules, such as those of (Langkilde-Geary, 2002; Carroll and Oepen, 2005), created semi-automatically (Belz, 2007) or, alternatively, extracted fully automatically from treebanks (Bangalore and Rambow, 2000; Nakanishi et al., 2005; Cahill and van Genabith, 2006).

Insofar as it is a broad coverage generator, which has been trained and tested on sections of the WSJ corpus, our generator is closer to the generators of (Bangalore and Rambow, 2000; Langkilde-Geary, 2002; Nakanishi et al., 2005) than to those designed for more restricted domains such as weather forecast (Belz, 2007) and air travel domains (Ratnaparkhi, 2000).

Another feature which characterises statistical generators is the probability model used to select the most probable sentence from among the space of all possible sentences licensed by the grammar. One generation technique is to first generate all possible sentences, storing them in a word lattice (Langkilde and Knight, 1998) or, alternatively, a generation forest, a packed represention of alternate trees proposed by the generator (Langkilde, 2000), and then select the most probable sequence of words via an n-gram language model.

Increasingly syntax-based information is being incorporated directly into the generation model. For example, Carroll and Oepen (2005) describe a sentence realisation process which uses a hand-crafted HPSG grammar to generate a generation forest. A selective unpacking algorithm allows the extraction of an *n*-best list of realisations where realisation ranking is based on a maximum entropy model. This unpacking algorithm is used in (Velldal and Oepen, 2005) to rank realisations with features defined over HPSG derivation trees. They achieved the best results when combining the tree-based model with an *n*-gram language model.

Nakanishi et al. (2005) describe a treebank-extracted HPSG-based chart generator. Importing techniques developed for HPSG parsing, they apply a log linear model to a packed representation of all alternative derivation trees for a given input. They found that a model which included syntactic information outperformed a bigram model as well as a combination of bigram and syntax model.

The probability model described in this paper also incorporates syntactic information, however, unlike the discriminative HPSG models just described, it is a generative history- and PCFG-based model. While Belz (2007) and Humphreys et al. (2001) mention the use of contextual features for the rules in their generation models, they do not provide details nor do they provide a formal probability model. To the best of our knowledge this is the first paper providing a probabilistic generative, history-based generation model.

3 Surface Realisation from f-Structures

Cahill and van Genabith (2006) present a probabilistic surface generation model for LFG (Kaplan, 1995). LFG is a constraint-based theory of grammar, which analyses strings in terms of c(onstituency)-structure and f(unctional)-structure (Figure 1). C-structure is defined in terms of CFGs, and f-structures are recursive attribute-value matrices which represent abstract syntactic functions (such as SUBJect, OBJect, OBLique, COMPlement (sentential), ADJ(N)unct), agreement, control, long-distance dependencies and some semantic information (e.g. tense, aspect).

C-structures and f-structures are related in a projection architecture in terms of a piecewise correspondence ϕ.[1] The correspondence is indicated in

[1] Our formalisation follows (Kaplan, 1995).

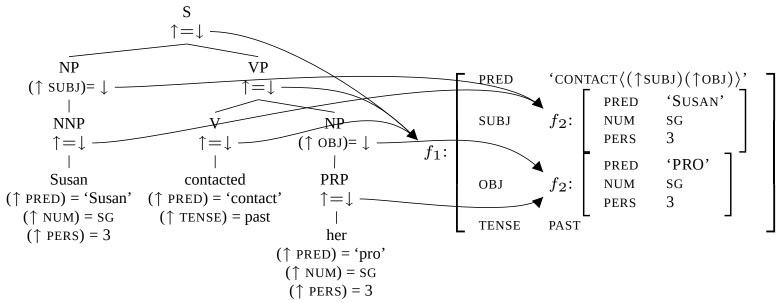


Figure 1: C- and f-structures with ϕ links for the sentence *Susan contacted her.*

terms of the curvy arrows pointing from c-structure nodes to f-structure components in Figure 1. Given a c-structure node n_i, the corresponding f-structure component f_j is $\phi(n_i)$. F-structures and the c-structure/f-structure correspondence are described in terms of functional annotations on c-structure nodes (CFG grammar rules). An equation of the form (↑F) = ↓ states that the f-structure associated with the mother of the current c-structure node (↑) has an attribute (grammatical function) (F), whose value is the f-structure of the current node (↓). The up-arrows and down-arrows are shorthand for $\phi(\mathrm{M}(n_i)) = \phi(n_i)$ where n_i is the c-structure node annotated with the equation.[2]

$$Tree_{best} := \text{argmax}_{\textit{Tree}} P(\textit{Tree}|\textit{F-Str}) \quad (1)$$

$$P(Tree|\textit{F-Str}) := \prod_{\substack{X \to Y \; in \; Tree \\ Feats = \{a_i | \exists v_j (\phi(X)) a_i = v_j\}}} P(X \to Y | X, Feats) \quad (2)$$

The generation model of (Cahill and van Genabith, 2006) maximises the probability of a tree given an f-structure (Eqn. 1), and the string generated is the yield of the highest probability tree. The generation process is guided by *purely* local information in the input f-structure: f-structure annotated CFG rules (LHS → RHS) are conditioned on their LHSs and on the set of features/attributes *Feats* = $\{a_i | \exists v_j \phi(\text{LHS}) a_i = v_j\}$[3] ϕ-linked to the LHS (Eqn. 2). Table 1 shows a generation grammar rule and conditioning features extracted from the example in Figure 1. The probability of a tree is decomposed into the product of the probabilities of the f-structure annotated rules (conditioned on the LHS and local *Feats*) contributing to the tree. Conditional probabilities are estimated using maximum likelihood estimation.

grammar rule	local conditioning features
S(↑=↓)⟶ NP(↑SUBJ=↓) VP(↑=↓)	S(↑=↓), {SUBJ,OBJ,PRED,TENSE}

Table 1: Example grammar rule (from Figure 1).

Cahill and van Genabith (2006) note that conditioning f-structure annotated generation rules on local features (Eqn. 2) can sometimes cause the model to make inappropriate choices. Consider the following scenario where in addition to the c-/f-structure in Figure 1, the training set contains the c-/f-structure displayed in Figure 2.

From Figures 1 and 2, the model learns (among others) the generation rules and conditional probabilities displayed in Tables 2 and 3.

F-Struct Feats	Grammar Rules	Prob
{SUBJ, OBJ, PRED}	S(↑=↓) ⟶ NP(↑SUBJ=↓) VP(↑=↓)	1
{SUBJ, OBJ, PRED}	VP(↑=↓) ⟶ V(↑=↓) NP(↑OBJ=↓)	1
{NUM, PER, GEN}	NP(↑SUBJ=↓) ⟶ NNP(↑=↓)	0.5
{NUM, PER, GEN}	NP(↑SUBJ=↓) ⟶ PRP(↑=↓)	0.5
{NUM, PER, GEN}	NP(↑OBJ=↓) ⟶ PRP(↑=↓)	1

Table 2: A sample of internal grammar rules extracted from Figures 1 and 2.

Given the input f-structure (for `She accepted`) in Figure 3, (and assuming suitable generation rules for intransitive VPs and `accepted`) the model would produce the inappropriate highest probability tree of Figure 4 with an incorrect case for the pronoun in subject position.

[2] M is the mother function on CFG tree nodes.

[3] In words, *Feats* is the set of top level features/attributes (those attributes a_i for which there is a value v_i) of the f-structure ϕ linked to the LHS.

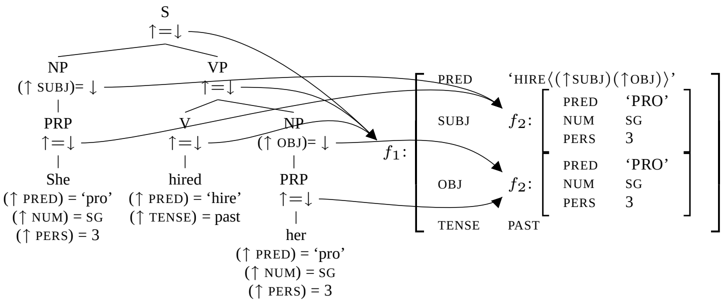


Figure 2: C- and f-structures with ϕ links for the sentence *She hired her.*

F-Struct Feats	Grammar Rules	Prob
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP(↑=↓) ⟶ she	0.33
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP(↑=↓) ⟶ her	0.66

Table 3: A sample of lexical item rules extracted from Figures 1 and 2.

$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{pro} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 3 \\ \text{GEND} & \text{fem} \end{array}\right] \\ \text{PRED} & \text{accept} \\ \text{TENSE} & \text{past} \end{array}\right]$$

Figure 3: Input f-structure for `She accepted`.

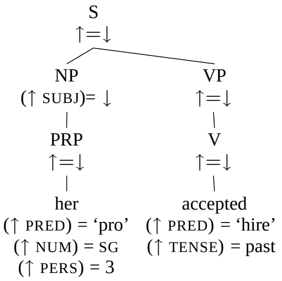


Figure 4: Inappropriate output: *her accepted.*

F-Struct Feats	Grammar Rules
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP-nom(↑=↓) ⟶ she
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP-acc(↑=↓) ⟶ her

Table 5: Lexical item rules with case markings

To solve the problem, Cahill and van Genabith (2006) apply an automatic generation grammar transformation to their training data: they automatically label CFG nodes with additional case information and the model now learns the new improved generation rules of Tables 4 and 5. Note how the additional case labelling subverts the problematic independence assumptions of the probability model and communicates the fact that a subject NP has to be realised as nominative case from the *S* ⟶ *NP-nom VP* production, via the intermediate *NP-nom* ⟶ *PRP-nom*, down to the lexical production *PRP-nom* ⟶ *she*. The labelling guarantees that, given the example f-structure in Figure 3, the model generates the correct string `she accepted`.

F-Struct Feats	Grammar Rules
{SUBJ, OBJ, PRED}	S(↑=↓) ⟶ NP-nom(↑SUBJ=↓) VP(↑=↓)
{SUBJ, OBJ, PRED}	VP(↑=↓) ⟶ V(↑=↓) NP-acc(↑OBJ=↓)
{NUM, PER, GEN}	NP-nom(↑SUBJ=↓) ⟶ PRP-nom(↑=↓)
{NUM, PER, GEN}	NP-nom(↑SUBJ=↓) ⟶ NNP-nom(↑=↓)
{NUM, PER, GEN}	NP-acc(↑OBJ=↓) ⟶ PRP-acc(↑=↓)

Table 4: Internal grammar rules with case markings.

4 A History-Based Generation Model

The automatic generation grammar transform presented in (Cahill and van Genabith, 2006) provides a solution to coarse-grained and (in fact) inappropriate independence assumptions in the basic generation model. However, there is a sense in which the proposed cure improves on the symptoms, but not the cause of the problem: it weakens independence assumptions by multiplying and hence increasing the specificity of conditioning CFG category labels. There is another option available to us, and that is the option we will explore in this paper: instead of applying a generation grammar transform, we will improve the f-structure-based conditioning of the generation rule probabilities. In the original model, rules are conditioned on *purely local* f-structure context: the set of features/attributes ϕ-linked to the LHS of a grammar rule. As a direct consequence of this, the conditioning (and hence the model) cannot not distinguish between NP, PRP and NNP rules appropriate to e.g. subject (SUBJ) or object contexts (OBJ) in a given input f-structure. However, the required information can easily be incorporated into the generation model by uniformly conditioning generation rules on their *parent* (*mother*) grammatical function, in addition to the local ϕ-linked feature set. This additional conditioning has the effect of making the choice of generation rules sensitive to the *history* of the generation process, and, we argue, provides a simpler, more uniform, general, intuitive and natural probabilistic generation model obviating the need for CFG-grammar transforms in the original proposal of (Cahill and van Genabith, 2006).

In the new model, each generation rule is now conditioned on the LHS rule CFG category, the set of features ϕ-linked to LHS *and* the parent grammatical function of the f-structure ϕ-linked to LHS. In a given c-/f-structure pair, for a CFG node n, the parent grammatical function of the f-structure ϕ-linked to n is that grammatical function GF, which, if we take the f-structure ϕ-linked to the mother M(n), and apply it to GF, returns the f-structure ϕ-linked to n: $(\phi(\mathrm{M}(n))\mathrm{GF}) = \phi(n)$.

The basic idea is best explained by way of an example. Consider again Figure 1. The mother grammatical function of the f-structure f_2 associated with node NP(↑SUBJ=↓) and its daughter NNP(↑=↓) (via the ↑=↓ functional annotation) is SUBJ, as $(\phi(\mathrm{M}(n_2))\mathrm{SUBJ}) = \phi(n_2)$, or equivalently $(f_1\mathrm{SUBJ}) = f_2$.

Given Figures 1 and 2 as training set, the improved model learns the generation rules (the mother grammatical function of the outermost f-structure is assumed to be a dummy TOP grammatical function) of Tables 6 and 7.

F-Struct Feats	Grammar Rules
{SUBJ, OBJ, PRED, **TOP**}	S(↑=↓) → NP(↑SUBJ=↓) VP(↑=↓)
{SUBJ, OBJ, PRED, **TOP**}	VP(↑=↓) → V(↑=↓) NP(↑OBJ=↓)
{NUM, PER, GEN, **SUBJ**}	NP(↑SUBJ=↓) → PRP(↑=↓)
{NUM, PER, GEN, **OBJ**}	NP(↑OBJ=↓) → PRP(↑=↓)
{NUM, PER, GEN, **SUBJ**}	NP(↑SUBJ=↓) → NNP(↑=↓)

Table 6: Grammar rules with extra feature extracted from F-Structures.

Note, that for our example the effect of the uniform additional conditioning on mother grammatical function has the same effect as the generation grammar transform of (Cahill and van Genabith, 2006), but without the need for the grammar transform. Given the input f-structure in Figure 3, the model will generate the correct string `she accepted`. In addition, uniform conditioning on mother grammatical function is more general than the case-phenomena specific generation grammar transform of (Cahill and van Genabith, 2006), in that it applies to each and every sub-part of a recursive input f-structure driving generation, making available relevant generation history (context) to guide local generation decisions.

F-Struct Feats	Grammar Rules
{PRED=PRO,NUM=SG PER=3, GEN=FEM, **SUBJ**}	PRP(↑=↓) → she
{PRED=PRO,NUM=SG PER=3, GEN=FEM, **OBJ**}	PRP(↑=↓) → her

Table 7: Lexical item rules.

The new history-based probabilistic generation model is defined as:

$$P(Tree|\textit{F-Str}) := \prod_{\substack{X \to Y \; in \; Tree \\ Feats = \{a_i | \exists v_j (\phi(X))a_i = v_j\} \\ (\phi(\mathrm{M}(X)))\mathbf{GF} = \phi(X)}} P(X \to Y | X, Feats, \mathbf{GF}) \tag{3}$$

Note that the new conditioning feature, the f-structure mother grammatical function, GF, is available from structure previously generated in the c-structure tree. As such, it is part of the *history* of the tree, i.e. it has already been generated in the top-down derivation of the tree. In this way, the generation model resembles history-based models for parsing (Black et al., 1992; Collins, 1999; Charniak, 2000). Unlike, say, the parent annotation for parsing of (Johnson, 1998) the parent GF feature for a particular node expansion is not merely extracted from the parent node in the c-structure tree, but is sometimes extracted from an ancestor node further up the c-structure tree via intervening ↑=↓ functional annotations.

Section 6 provides evaluation results for the new model on section 23 of the Penn treebank.

5 Multi-Word Units

In another effort to improve generator accuracy over the baseline model we explored the use of multi-word units in generation. We expect that the identification of MWUs may be useful in imposing word-order constraints and reducing the complexity of the generation task. Take, for example, the following

$$\begin{bmatrix} \textsc{app} & \begin{bmatrix} \textsc{adjunct} & \begin{bmatrix} \textsc{pred} & \text{`New'} \\ \textsc{num} & \text{sg} \\ \textsc{pers} & 3 \end{bmatrix} \\ \textsc{pred}\ \text{`York'} & \\ \textsc{num}\ \text{sg} & \\ \textsc{pers}\ 3 & \end{bmatrix} \end{bmatrix} \begin{bmatrix} \textsc{app} & \begin{bmatrix} \textsc{pred} & \text{`New\_York'} \\ \textsc{num} & \text{sg} \\ \textsc{pers} & 3 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \textsc{app} & \begin{bmatrix} \textsc{adjunct} & \begin{bmatrix} \textsc{pred} & \text{`New'/NE1\_1} \\ \textsc{num} & \text{sg} \\ \textsc{pers} & 3 \end{bmatrix} \\ \textsc{pred}\ \text{`York'/NE1\_2} & \\ \textsc{num}\ \text{sg} & \\ \textsc{pers}\ 3 & \end{bmatrix} \end{bmatrix}$$

Figure 5: Three different f-structure formats. From left to right: the original f-structure format; the MWU chunk format; the MWU mark-up format.

two sentences which show the gold version of a sentence followed by the version of the sentence produced by the generator:

Gold *By this time , it was 4:30 a.m. in New York , and Mr. Smith fielded a call from a New York customer wanting an opinion on the British stock market , which had been having troubles of its own even before Friday 's New York market break .*

Test *By this time , in New York , it was 4:30 a.m. , and Mr. Smith fielded a call from* **New** *a customer* **York** *, wanting an opinion on the market British stock which had been having troubles of its own even before Friday 's New York market break .*

The gold version of the sentence contains a multi-word unit, *New York*, which appears fragmented in the generator output. If multi-word units were either treated as one token throughout the generation process, or, alternatively, if a constraint were imposed on the generator such that multi-word units were always generated in the correct order, then this should help improve generation accuracy. In Section 5.1 we describe the various techniques that were used to incorporate multi-word units into the generation process and in 5.2 we detail the different types and sources of multi-word unit used in the experiments. Section 6 provides evaluation results on test and development sets from the WSJ treebank.

5.1 Incorporating MWUs into the Generation Process

We carried out three types of experiment which, in different ways, enabled the generation process to respect the restrictions on word-order provided by multi-word units. For the first experiments (type 1), the WSJ treebank *training* and *test* data were altered so that multi-word units are concatenated into single words (for example, *New York* becomes *New_York*). As in (Cahill and van Genabith, 2006) f-structures are generated from the (now altered) treebank and from this data, along with the treebank trees, the PCFG-based grammar, which is used for training the generation model, is extracted. Similarly, the f-structures for the test and development sets are created from Penn Treebank trees which have been modified so that multi-word units form single units. The leftmost and middle f-structures in Figure 5 show an example of an original f-structure format and a named-entity chunked format, respectively. Strings output by the generator are then post-processed so that the concatenated word sequences are converted back into single words.

In the second experiment (type 2) only the *test* data was altered with no concatenation of MWUs carried out on the training data.

In the final experiments (type 3), instead of concatenating named entities, a constraint is introduced to the generation algorithm which penalises the generation of sequences of words which violate the internal word order of named entities. The input is marked-up in such a way that, although named entities are no longer chunked together to form single words, the algorithm can read which items are part of named entities. See the rightmost f-structure in Figure 5 for an example of an f-structure marked-up in this way. The tag *NE1_1*, for example, indicates that the sub-f-structure is part of a named identity with id number 1 and that the item corresponds to the first word of the named entity. The baseline generation algorithm, following Kay (1996)'s work on chart generation, already contains the hard constraint that when combining two chart edges they must cover disjoint sets of words. We added an additional constraint which prevents edges from being combined if this would result in the generation of a string which contained a named entity which was either incomplete or where the words in the named entity were generated in the wrong order.

5.2 Types of MWUs used in Experiments

We carry out experiments with multi-word units from three different sources. First, we use the output of the maximum entropy-based named entity recognition system of (Chieu and Ng, 2003). This system identifies four types of named entity: person, organisation, location, and miscellaneous. Additionally we use a dictionary of candidate multi-word expressions based on a list from the Stanford Multiword Expression Project[4]. Finally, we also carry out experiments with multi-word units extracted from the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005). This supplements the Penn WSJ treebank's one million words of syntax-annotated Wall Street Journal text with additional annotations of 23 named entity types, including nominal-type named entities such as person, organisation, location, etc. as well as numeric types such as date, time, quantity and money. Since the BBN corpus data is very comprehensive and is hand-annotated we take this be be a gold standard, representing an upper bound for any gains that might be made by identifying complex named entities in our experiments.[5] Table 8 gives examples of the various types of MWUs identified by the three sources.

For our purposes we are not concerned with the distinctions between different types of named entities; we are merely exploiting the fact that they may be treated as atomic units in the generation model. In all cases we disregard multi-word units that cross the original syntactic bracketing of the WSJ treebank. An overview of the various types of multi-word units used in our experiments is presented in Table 9.

MWU type	Examples
Names	*Martha Matthews* *Yoshio Hatakeyama*
Organisations	*Rolls-Royce Motor Cars Inc.* *Washington State University*
Locations	*New York City* *New Zealand*
Time expressions	*October 19th* *two years ago* *the 21st century*
Quantities	*$2.7 million to $3 million* *about 25 %* *60 mph*
Prepositional expressions	*in fact* *at the time* *on average*

Table 8: Examples of some of the types of MWU from the three different sources.

	average number	average length
(Chieu and Ng, 2003)	0.61	2.40
Stanford MWE Project	0.10	2.48
BBN Corpus	1.15	2.66

Table 9: Average number of MWUs per sentence and average MWU length in the WSJ treebank grouped by MWU source.

6 Experimental Evaluation

All experiments were carried out on the WSJ treebank with sections 02-21 for training, section 24 for development and section 23 for final test results. The LFG annotation algorithm of (Cahill et al., 2004) was used to produce the f-structures for development, test and training sets.

Table 10 shows the final results for section 23. For each test we present BLEU score results as well as String Edit Distance and coverage. We measure statistical significance using two different tests. First we use a bootstrap resampling method, popular for machine translation evaluations, to measure the significance of improvements in BLEU scores, with a resampling rate of 1000.[6] We also calculated the significance of an increase in String Edit Distance by carrying out a paired t-test on the mean difference of the String Edit Distance scores. In Table 10, $\gg$ means significant at level 0.005. $>$ means significant at level 0.05.

In Table 10, *Baseline* gives the results of the generation algorithm of (Cahill and van Genabith, 2006). *HB Model* refers to the improved model with the increased history context, as described in Section 4. The results, where for example the BLEU score rises from 66.52 to 67.24, show that even increasing the conditioning context by a limited

[4] mwe.stanford.edu

[5] Although it is possible there are other types of MWUs that may be more suitable to the task than the named entities identified by BBN, so further gains might be possible.

[6] Scripts for running the bootstrapping method carried out in our evaluation are available for download at projectile.is.cs.cmu.edu/research/public/tools/bootStrap/tutorial.htm

	Section 23 (2416 sentences)				
Model	BLEU	StringEd	Coverage	BLEU Bootstrap Signif	StringEd Paired T-Test
1. Baseline	66.52	68.69	98.18		
2. HB Model	67.24	69.89	99.88	$\gg 1$	$\gg 1$
3. +MWU Best Automatic	67.81	70.36	99.92	$\gg 2$	$\gg 2$
4. MWU BBN	68.82	70.92	99.96	$\gg 3$	> 3

Table 10: Results on Section 23 for all sentence lengths.

amount increases the accuracy of the system significantly for both BLEU and String Edit Distance. In addition, coverage goes up from 98.18% to 99.88%.

+MWU Best Automatic displays our best results using automatically identified named entities. These were achieved using experiment type 2, described in Section 5, with the MWUs produced by (Chieu and Ng, 2003). Results displayed in Table 10 up to this point are cumulative. The final row in Table 10, *MWU BBN*, shows the best results with BBN MWUs: the history-based model with BBN multiword units incorporated in a type 1 experiment.

We now discuss the various MWU experiments in more detail. See Table 11 for a breakdown of the MWU experiment results on the development set, WSJ section 24. Our baseline for these experiments is the history-based generator presented in Section 4. For each experiment type described in Section 5.1 we ran three experiments, varying the source of MWUs. First, MWUs came from the automatic NE recogniser of (Chieu and Ng, 2003), then we added the MWUs from the Stanford list and finally we ran tests with MWUs extracted from the BBN corpus.

Our first set of experiments (type 1), where both training data and development set data were MWU-chunked, produced the worst results for the automatically chunked MWUs. BLEU score accuracy actually decreased for the automatically chunked MWU experiments. In an error analysis of type 1 experiments with (Chieu and Ng, 2003) concatenated MWUs, we inspected those sentences where accuracy had decreased from the baseline. We found that for over half (51.5%) of these sentences, the input f-structures contained no multi-word units at all. The problem for these sentences therefore lay with the probabilistic grammar extracted from the MWU-chunked training data. When the source of MWU for the type 1 experiments was the BBN, however, accuracy improved significantly over the baseline and the result is the highest accuracy achieved over all experiment types. One possible reason for the low accuracy scores in the type 1 experiments with the (Chieu and Ng, 2003) MWU chunked data could be noisy MWUs which negatively affect the grammar. For example, the named entity recogniser of (Chieu and Ng, 2003) achieves an accuracy of 88.3% on section 23 of the Penn Treebank.

In order to avoid changing the grammar through concatenation of MWU components (as in experiment type 1) and thus risking side-effects which cause some heretofore likely constructions become less likely and vice versa, we ran the next set of experiments (type 2) which leave the original grammar intact and alter the input f-structures only. These experiments were more successful overall and we achieved an improvement over the baseline for both BLEU and String Edit Distance scores with all MWU types. As can be seen from Table 11 the best score for automatically chunked MWUs are with the (Chieu and Ng, 2003) MWUs. Accuracy decreases marginally when we added the Stanford MWUs. In our final set of experiments (type 3) although the accuracy for all three types of MWUs improves over the baseline, accuracy is a little below the type 2 experiments.

It is difficult to compare sentence generators since the information contained in the input varies greatly between systems, systems are evaluated on different test sets and coverage also varies considerably. In order to compare our system with those of (Nakanishi et al., 2005) and (Langkilde-Geary, 2002) we report our best results with automatically acquired MWUs for sentences of ≤ 20 words in length on section 23: our system gets coverage of 100% and a BLEU score of 71.39. For the same test set Nakanishi et al. (2005) achieved coverage of 90.75 and a BLEU score of 77.33. Langkilde-Geary (2002) re-

		Section 24 (1346 sentences)		
Model	MWUs	BLEU	StringEd	Coverage
HB Model		65.85	69.93	99.93
type 1	(Chieu and Ng, 2003)	65.81	70.34	99.93
(training and test data chunked)	+Stanford MWEs	64.81	69.67	99.93
	BBN	67.24	71.46	99.93
type 2	(Chieu and Ng, 2003)	66.37	70.26	99.93
(test data chunked)	+Stanford MWEs	66.28	70.21	99.93
	BBN	66.84	70.74	99.93
type 3	(Chieu and Ng, 2003)	66.30	70.12	100
(internal generation constraint)	+Stanford MWEs	66.07	70.02	99.93
	BBN	66.45	70.14	99.93

Table 11: Results on Section 24, all sentence lengths.

ports 82.7% coverage and a BLEU score of 75.7% on the same test set with the ‘permute,no dir’ type input. Langkilde-Geary (2002) report results for experiments with varying levels of linguistic detail in the input given to the generator. As with Nakanishi et al. (2005) we find the ‘permute,no dir’ type of input is most comparable to the level of information contained in our input f-structures. Finally, the symbolic generator of Callaway (2003) reports a Simple String Accuracy score of 88.84 and coverage of 98.7% on section 23 for all sentence lengths.

7 Conclusion and Future Work

We have presented techniques which improve the accuracy of an already state-of-art surface generation model. We found that a history-based model that increases conditioning context in PCFG style rules by simply including the grammatical function of the f-structure parent, improves generator accuracy. In the future we will experiment with increasing conditioning context further and using more sophisticated smoothing techniques to avoid sparse data problems when conditioning is increased.

We have also demonstrated that automatically acquired multi-word units can bring about moderate, but significant, improvements in generator accuracy. For automatically acquired MWUs, we found that this could best be achieved by concatenating input items when generating the f-structure input to the generator, while training the input generation grammar on the original (i.e. non-MWU concatenated) sections of the treebank. Relying on the BBN corpus as a source of multi-word units, we gave an upper bound to the potential usefulness of multi-word units in generation and showed that automatically acquired multi-word units, encouragingly, give results not far below the upper bound.

References

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th COLING*.

Anja Belz. 2007. Probabilistic generation of wether forecast texts. In *Proceedings of NAACL-HLT*.

Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceeding of the 5th DARPA Speech and Language Workshop*.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 44th ACL*.

Aoife Cahill, Michael Burke, Ruth O’Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd ACL*.

Charles B. Callaway. 2003. Evaluating coverage for large symbolic NLG grammars. In *In Proceedings of the 18th IJCAI*.

John A. Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of IJCNLP*.

Eugene Charniak. 2000. A maximum entropy-inspired parser. In *Proceedings of the 1st NAACL*.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the CoNLL*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Kevin Humphreys, Mike Calcagno, and David Weise. 2001. Reusing a statistical language model for generation. In *Proceedings of the 8th European Workshop on Natural Language Generation (EWNLG)*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24.

Ronald M. Kaplan and Tracy Holloway King. 2003. Low-level mark-up and large-scale LFG grammar processing. In *Proceedings of the Lexical Functional Grammar Conference*.

Ron Kaplan. 1995. The formal architecture of lexical-functional grammar. In Dalrymple, Kaplan, Maxwell, and Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*, pages 7–27. CSLI Publications.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th ACL*.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (ACL-COLING)*.

Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 2nd INLG*.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st NAACL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th IWPT*.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. In *Workshop on Methodologies and Evaluation of Multiword Units in Real-World Applications*.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of the 1st NAACL*.

Stefan Riezler and John T. Maxwell. 2006. Grammatical machine translation. In *Proceedings of the 6th NAACL*.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the 3rd NAACL*.

Erik Velldal and Stephan Oepen. 2005. Maximum entropy models for realization ranking. In *Proceedings of the MT-Summit*.

Ralph Weischedel and Ada Brunstein, 2005. *BBN pronoun coreference and entity type corpus*. Technical Report.

Hierarchical System Combination for Machine Translation

Fei Huang
IBM T.J. Watson Research Center
Yorktown Heights, NY 10562
huangfe@us.ibm.com

Kishore Papineni *
Yahoo! Research
New York, NY 10011
kpapi@yahoo-inc.com

Abstract

Given multiple translations of the same source sentence, how to combine them to produce a translation that is better than any single system output? We propose a hierarchical system combination framework for machine translation. This framework integrates multiple MT systems' output at the word-, phrase- and sentence- levels. By boosting common word and phrase translation pairs, pruning unused phrases, and exploring decoding paths adopted by other MT systems, this framework achieves better translation quality with much less re-decoding time. The full sentence translation hypotheses from multiple systems are additionally selected based on N-gram language models trained on word/word-POS mixed stream, which further improves the translation quality. We consistently observed significant improvements on several test sets in multiple languages covering different genres.

1 Introduction

Many machine translation (MT) frameworks have been developed, including rule-based transfer MT, corpus-based MT (statistical MT and example-based MT), syntax-based MT and the hybrid, statistical MT augmented with syntactic structures. Different MT paradigms have their strengths and weaknesses. Systems adopting the same framework usually produce different translations for the same input, due to their differences in training data, preprocessing, alignment and decoding strategies. It is beneficial to design a framework that combines the decoding strategies of multiple systems as well as their outputs and produces translations better than any single system output. More recently, within the GALE[1] project, multiple MT systems have been developed in each consortium, thus system combination becomes more important.

Traditionally, system combination has been conducted in two ways: glass-box combination and black-box combination. In the glass-box combination, each MT system provides detailed decoding information, such as word and phrase translation pairs and decoding lattices. For example, in the multi-engine machine translation system (Nirenburg and Frederking, 1994), target language phrases from each system and their corresponding source phrases are recorded in a chart structure, together with their confidence scores. A chart-walk algorithm is used to select the best translation from the chart. To combine words and phrases from multiple systems, it is preferable that all the systems adopt similar preprocessing strategies.

In the black-box combination, individual MT systems only output their top-N translation hypotheses without decoding details. This is particularly appealing when combining the translation outputs from COTS MT systems. The final translation may be selected by voted language models and appropriate confidence rescaling schemes ((Tidhar and Kuss-

[*] This work was done when the author was at IBM Research.

[1] http://www.darpa.mil/ipto/programs/gale/index.htm

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 277–286, Prague, June 2007. ©2007 Association for Computational Linguistics

ner, 2000) and (Nomoto, 2004)). (Mellebeek et al., 2006) decomposes source sentences into meaningful constituents, translates them with component MT systems, then selects the best segment translation and combine them based on majority voting, language models and confidence scores.

(Jayaraman and Lavie, 2005) proposed another black-box system combination strategy. Given single top-one translation outputs from multiple MT systems, their approach reconstructs a phrase lattice by aligning words from different MT hypotheses. The alignment is based on the surface form of individual words, their stems (after morphology analysis) and part-of-speech (POS) tags. Aligned words are connected via edges. The algorithm finds the best alignment that minimizes the number of crossing edges. Finally the system generates a new translation by searching the lattice based on alignment information, each system's confidence scores and a language model score. (Matusov et al., 2006) and (Rosti et al., 2007) constructed a confusion network from multiple MT hypotheses, and a consensus translation is selected by redecoding the lattice with arc costs and confidence scores.

In this paper, we introduce our hierarchical system combination strategy. This approach allows combination on word, phrase and sentence levels. Similar to glass-box combination, each MT system provides detailed information about the translation process, such as which source word(s) generates which target word(s) in what order. Such information can be combined with existing word and phrase translation tables, and the augmented phrase table will be significantly pruned according to reliable MT hypotheses. We select an MT system to retranslate the test sentences with the refined models, and encourage search along decoding paths adopted by other MT systems. Thanks to the refined translation models, this approach produces better translations with a much shorter re-decoding time. As in the black-box combination, we select full sentence translation hypotheses from multiple system outputs based on n-gram language models. This hierarchical system combination strategy avoids problems like translation output alignment and confidence score normalization. It seamlessly integrates detailed decoding information and translation hypotheses from multiple MT engines, and produces better translations in an efficient manner. Empirical studies in a later section show that this algorithm improves MT quality by 2.4 BLEU point over the best baseline decoder, with a 1.4 TER reduction. We also observed consistent improvements on several evaluation test sets in multiple languages covering different genres by combining several state-of-the-art MT systems.

The rest of the paper is organized as follows: In section 2, we briefly introduce several baseline MT systems whose outputs are used in the system combination. In section 3, we present the proposed hierarchical system combination framework. We will describe word and phrase combination and pruning, decoding path imitation and sentence translation selection. We show our experimental results in section 4 and conclusions in section 5.

2 Baseline MT System Overview

In our experiments, we take the translation outputs from multiple MT systems. These include phrase-based statistical MT systems (Al-Onaizan and Papineni, 2006) (Block) and (Hewavitharana et al., 2005) (CMU_SMT) , a direct translation model (DTM) system (Ittycheriah and Roukos, 2007) and a hierarchical phrased-based MT system (Hiero) (Chiang, 2005). Different translation frameworks are adopted by different decoders: the DTM decoder combines different features (source words, morphemes and POS tags, target words and POS tags) in a maximum entropy framework. These features are integrated with a phrase translation table for flexible distortion model and word selection. The CMU_SMT decoder extracts testset-specific bilingual phrases on the fly with PESA algorithm. The Hiero system extracts context-free grammar rules for long range constituent reordering.

We select the IBM block decoder to re-translate the test set for glass-box system combination. This system is a multi-stack, multi-beam search decoder. Given a source sentence, the decoder tries to find the translation hypothesis with the minimum translation cost. The overall cost is the log-linear combination of different feature functions, such as translation model cost, language model cost, distortion cost and sentence length cost. The translation cost between a phrase translation pair (f, e) is defined as

$$TM(e, f) = \sum_i \lambda_i \phi(i) \quad (1)$$

where feature cost functions $\phi(i)$ includes:

$-\log p(f|e)$, a target-to-source word translation cost, calculated based on unnormalized IBM model1 cost (Brown et al., 1994);

$$p(f|e) = \prod_j \sum_i t(f_j|e_i) \quad (2)$$

where $t(f_j|e_i)$ is the word translation probabilities, estimated based on word alignment frequencies over all the training data. i and j are word positions in target and source phrases.

$-\log p(e|f)$, a source-to-target word translation cost, calculated similar to $-\log p(f|e)$;

$S(e, f)$, a phrase translation cost estimated according to their relative alignment frequency in the bilingual training data,

$$S(e, f) = -\log P(e|f) = -\log \frac{C(f, e)}{C(f)}. \quad (3)$$

λ's in Equation 1 are the weights of different feature functions, learned to maximize development set BLEU scores using a method similar to (Och, 2003).

The SMT system is trained with testset-specific training data. This is not cheating. Given a test set, from a large bilingual corpora we select parallel sentence pairs covering n-grams from source sentences. Phrase translation pairs are extracted from the subsampled alignments. This not only reduces the size of the phrase table, but also improves topic relevancy of the extracted phrase pairs. As a results, it improves both the efficiency and the performance of machine translation.

3 Hierarchical System Combination Framework

The overall system combination framework is shown in Figure 1. The source text is translated by multiple baseline MT systems. Each system produces both top-one translation hypothesis as well as phrase pairs and decoding path during translation. The information is shared through a common XML file format, as shown in Figure 2. It demonstrates how a source sentence is segmented into a sequence of phrases, the order and translation of each source phrase as well as the translation scores, and a vector of feature scores for the whole test sentence. Such XML files are generated by all the systems when they translate the source test set.

We collect phrase translation pairs from each decoder's output. Within each phrase pair, we identify word alignment and estimate word translation probabilities. We combine the testset-specific word translation model with a general model. We augment the baseline phrase table with phrase translation pairs extracted from system outputs, then prune the table with translation hypotheses. We retranslate the source text using the block decoder with updated word and phrase translation models. Additionally, to take advantage of flexible reordering strategies of other decoders, we develop a word order cost function to reinforce search along decoding paths adopted by other decoders. With the refined translation models and focused search space, the block decoder efficiently produces a better translation output. Finally, the sentence hypothesis selection module selects the best translation from each systems' top-one outputs based on language model scores. Note that the hypothesis selection module does not require detailed decoding information, thus can take in any MT systems' outputs.

3.1 Word Translation Combination

The baseline word translation model is too general for the given test set. Our goal is to construct a testset-specific word translation model, combine it with the general model to boost consensus word translations. Bilingual phrase translation pairs are read from each system-generated XML file. Word alignments are identified within a phrase pair based on IBM Model-1 probabilities. As the phrase pairs are typically short, word alignments are quite accurate. We collect word alignment counts from the whole test set translation, and estimate both source-to-target and target-to-source word translation probabilities. We combine such testset-specific translation model with the general model.

$$t''(e|f) = \gamma t'(e|f) + (1 - \gamma) t(e|f); \quad (4)$$

where $t'(e|f)$ is the testset-specific source-to-target word translation probability, and $t(e|f)$ is the prob-

```
    <tr engine="XXX">
    <s id="0"> <w> اردوغان </w><w> يؤكد </w><w> بان </w><w> تركيا </w><w> سترفض
</w><w> اي </w><w> ضغوطات </w><w> لحثها </w><w> علي </w><w> الاعتراف </w><w>
بقبرص </w></s>
    <hyp r="0" c="2.15357">
     <t>
    <p al="0-0" cost="0.0603734"> erdogan </p>
    <p al="1-1" cost="0.367276"> emphasized </p>
    <p al="2-2" cost="0.128066"> that </p>
    <p al="3-3" cost="0.0179338"> turkey </p>
    <p al="4-5" cost="0.379862"> would reject any </p>
    <p al="6-6" cost="0.221536"> pressure </p>
    <p al="7-7" cost="0.228264"> to urge them </p>
    <p al="8-8" cost="0.132242"> to</p>
    <p al="9-9" cost="0.113983"> recognize </p>
    <p al="10-10" cost="0.133359"> Cyprus </p>
    </t>
    <sco>
    19.6796 8.40107 0.333514 0.00568583 0.223554 0 0.352681 0.01 -0.616 0.009 0.182052
    </sco>
    </hyp>
    </tr>
```

Figure 2: Sample XML file format. This includes a source sentence (segmented as a sequence of source phrases), their translations as well as a vector of feature scores (language model scores, translation model scores, distortion model scores and a sentence length score).

ability from general model. γ is the linear combination weight, and is set according to the confidence on the quality of system outputs. In our experiments, we set γ to be 0.8. We combine both source-to-target and target-to-source word translation models, and update the word translation costs, $-\log p(e|f)$ and $-\log p(f|e)$, accordingly.

3.2 Phrase Translation Combination and Pruning

Phrase translation pairs can be combined in two different ways. We may collect and merge testset-specific phrase translation tables from each system, if they are available. Essentially, this is similar to combining the training data of multiple MT systems. The new phrase translation probability is calculated according to the updated phrase alignment frequencies:

$$P'(e|f) = \frac{C_b(f,e) + \sum \alpha_m C_m(f,e)}{C_b(f) + \sum \alpha_m C_m(f)}, \qquad (5)$$

where C_b is the phrase pair count from the baseline block decoder, and C_m is the count from other MT systems. α_m is a system-specific linear combination weight. If not all the phrase tables are available, we collect phrase translation pairs from system outputs, and merge them with C_b. In such case, we may adjust α to balance the small counts from system outputs and large counts from C_b.

The corresponding phrase translation cost is updated as

$$S'(e,f) = -\log P'(e|f). \qquad (6)$$

Another phrase combination strategy works on the sentence level. This strategy relies on the consensus of different MT systems when translating the same source sentence. It collects phrase translation pairs used by different MT systems to translate the same sentence. Similarly, it boosts common phrase pairs that are selected by multiple decoders.

$$S''(e,f) = \frac{\beta}{|C(f,e)|} \times S'(e,f), \qquad (7)$$

where β is a boosting factor, $0 < \beta \leq 1$. $|C(f,e)|$ is the number of systems that use phrase pair (f,e) to translate the input sentence. A phrase translation pair selected by multiple systems is more likely a good translation, thus costs less.

The combined phrase table contains multiple translations for each source phrase. Many of them are unlikely translations given the context. These phrase pairs produce low-quality partial hypotheses during hypothesis expansion, incur unnecessary model cost calculation and larger search space, and reduce the translation efficiency. More importantly, the translation probabilities of correct phrase pairs are reduced as some probability mass is distributed among incorrect phrase pairs. As a result, good phrase pairs may not be selected in the final translation.

Oracle experiments show that if we prune the phrase table and only keep phrases that appear in the reference translations, we can improve the translation quality by 10 BLEU points. This shows the potential gain by appropriate phrase pruning. We developed a phrase pruning technique based on self-training. This approach reinforces phrase translations learned from MT system output. Assuming we have reasonable first-pass translation outputs, we only keep phrase pairs whose target phrase is covered by existing system translations. These phrase pairs include those selected in the final translations, as well as their combinations or sub-phrases. As a result, the size of the phrase table is reduced by 80-90%, and the re-decoding time is reduced by 80%. Because correct phrase translations are assigned higher probabilities, it generates better translations with higher BLEU scores.

3.3 Decoding Path Imitation

Because of different reordering models, words in the source sentence can be translated in different orders. The block decoder has local reordering capability that allows source words within a given window to jump forward or backward with a certain cost. The DTM decoder takes similar reordering strategy, with some variants like dynamic window width depending on the POS tag of the current source word. The Hiero system allows for long range constituent reordering based on context-free grammar rules. To combine different reordering strategies from various decoders, we developed a reordering cost function that encourages search along decoding paths adopted by other decoders.

From each system's XML file, we identify the order of translating source words based on word alignment information. For example, given the following hypothesis path,

<p al="0-1"> izzat ibrahim </p> <p al="2-2"> receives </p> <p al="3-4"> an economic official </p> <p al="5-6"> in </p> <p al="7-7"> baghdad </p>

We find the source phrase containing words [0,1] is first translated into a target phrase "*izzat ibrahim*", which is followed by the translation from source word 2 to a single target word "*receives*", etc.. We identify the word alignment within the phrase translation pairs based on IBM model-1 scores. As a result, we get the following source word translation sequence from the above hypothesis (note: source word 5 is translated as NULL):

$0 < 1 < 2 < 4 < 3 < 6 < 7$

Such decoding sequence determines the translation order between any source word pairs, e.g., word 4 should be translated before word 3, 6 and 7. We collect such ordered word pairs from all system outputs' paths. When re-translating the source sentence, for each partially expanded decoding path, we compute the ratio of word pairs that satisfy such ordering constraints[2].

Specifically, given a partially expanded path $P = \{s_1 < s_2 < \cdots < s_m\}$, word pair $(s_i < s_j)$ implies s_i is translated before s_j. If word pair $(s_i < s_j)$ is covered by a full decoding path Q (from other system outputs), we denote the relationship as $(s_i < s_j) \in Q$.

For any ordered word pair $(s_i < s_j) \in P$, we define its matching ratio as the percentage of full decoding paths that cover it:

$$R(s_i < s_j) = \frac{|Q|}{N}, \{Q|(s_i < s_j) \in Q\} \qquad (8)$$

where N is the total number of full decoding paths.

We define the path matching cost function:

$$L(P) = -\log \frac{\sum_{\forall(s_i<s_j)\in P} R(s_i < s_j)}{\sum_{\forall(s_i<s_j)\in P} 1} \qquad (9)$$

The denominator is the total number of ordered word pairs in path P. As a result, partial paths are boosted if they take similar source word translation orders as other system outputs. This cost function is multiplied with a manually tuned model weight before integrating into the log-linear cost model framework.

[2] We set no constraints for source words that are translated into NULL.

3.4 Sentence Hypothesis Selection

The sentence hypothesis selection module only takes the final translation outputs from individual systems, including the output from the glass-box combination. For each input source sentence, it selects the "optimal" system output based on certain feature functions.

We experiment with two feature functions. One is a typical 5-gram word language model (LM). The optimal translation output E' is selected among the top-one hypothesis from all the systems according to their LM scores. Let e_i be a word in sentence E:

$$\begin{aligned} E' &= \arg\min_E -\log P_{5glm}(E) \quad (10) \\ &= \arg\min_E \sum_i -\log p(e_i|e_{i-4}^{i-1}), \end{aligned}$$

where e_{i-4}^{i-1} is the n-gram history, $(e_{i-4}, e_{i-3}, e_{i-2}, e_{i-1})$.

Another feature function is based on the 5-gram LM score calculated on the mixed stream of word and POS tags of the translation output. We run POS tagging on the translation hypotheses. We keep the word identities of top N frequent words (N=1000 in our experiments), and the remaining words are replaced with their POS tags. As a result, the mixed stream is like a skeleton of the original sentence, as shown in Figure 3.

With this model, the optimal translation output E^* is selected based on the following formula:

$$\begin{aligned} E^* &= \arg\min_E -\log P_{wplm}(E) \quad (11) \\ &= \arg\min_E \sum_i -\log p(T(e_i)|T(e)_{i-4}^{i-1}) \end{aligned}$$

where the mixed stream token $T(e) = e$ when $e \leq N$, and $T(e) = POS(e)$ when $e > N$. Similar to a class-based LM, this model is less prone to data sparseness problems.

4 Experiments

We experiment with different system combination strategies on the NIST 2003 Arabic-English MT evaluation test set. Testset-specific bilingual data are subsampled, which include 260K sentence pairs, 10.8M Arabic words and 13.5M English words. We report case-sensitive BLEU (Papineni et al., 2001) and TER (Snover et al., 2006) as the MT evaluation metrics. We evaluate the translation quality of different combination strategies:

	BLEUr4n4c	TER
sys1	0.5323	43.11
sys4	0.4742	46.35
Tstcom	0.5429	42.64
Tstcom+Sentcom	0.5466	42.32
Tstcom+Sentcom+Prune	0.5505	42.21

Table 1: Translation results with phrase combination and pruning.

- **WdCom:** Combine testset-specific word translation model with the baseline model, as described in section 3.1.
- **PhrCom:** Combine and prune phrase translation tables from all systems, as described in section 3.2. This include testset-specific phrase table combination (**Tstcom**), sentence level phrase combination (**Sentcom**) and phrase pruning based on translation hypotheses (**Prune**).
- **Path:** Encourage search along the decoding paths adopted by other systems via path matching cost function, as described in section 3.3.
- **SenSel:** Select whole sentence translation hypothesis among all systems' top-one outputs based on N-gram language models trained on word stream (**word**) and word-POS mixed stream(**wdpos).**

Table 1 shows the improvement by combining phrase tables from multiple MT systems using different combination strategies. We only show the highest and lowest baseline system scores. By combining testset-specific phrase translation tables (**Tstcom**), we achieved 1.0 BLEU improvement and 0.5 TER reduction. Sentence-level phrase combination and pruning additionally improve the BLEU score by 0.7 point and reduce TER by 0.4 percent.

Table 2 shows the improvement with different sentence translation hypothesis selection approaches. The word-based LM is trained with about 1.75G words from newswire text. A distributed

	BLEUr4n4c	TER
sys1	0.5323	43.11
sys2	0.5320	43.06
SentSel-word:	0.5354	42.56
SentSel-wpmix:	0.5380	43.06

Table 2: Translation results with different sentence hypothesis selection strategies.

	BLEUr4n4c	TER
sys1	0.5323	43.11
sys2	0.5320	43.06
sys3	0.4922	46.03
sys4	0.4742	46.35
WdCom	0.5339	42.60
WdCom+PhrCom	0.5528	41.98
WdCom+PhrCom+Path	0.5543	41.75
WdCom+PhrCom+Path+SenSel	0.5565	41.59

Table 3: Translation results with hierarchical system combination strategy.

	BLEUr4n4c	TER
sys1	0.3205	60.48
sys2	0.3057	59.99
sys3	0.2787	64.46
sys4	0.2823	59.19
sys5	0.3028	62.16
syscom	0.3409	58.89

Table 4: System combination results on Chinese-English translation.

	BLEUr1n4c	TER
sys1	0.1261	71.70
sys2	0.1307	77.52
sys3	0.1282	70.82
sys4	0.1259	70.20
syscom	0.1386	69.23

Table 5: System combination results for Arabic-English web log translation.

large-scale language model architecture is developed to handle such large training corpora[3], as described in (Emami et al., 2007). The word-based LM shows both improvement in BLEU scores and error reduction in TER. On the other hand, even though the word-POS LM is trained with much less data (about 136M words), it improves BLEU score more effectively, though there is no change in TER.

Table 3 shows the improvements from hierarchical system combination strategy. We find that word-based translation combination improves the baseline block decoder by 0.16 BLEU point and reduce TER by 0.5 point. Phrase-based translation combination (including phrase table combination, sentence-level phrase combination and phrase pruning) further improves the BLEU score by 1.9 point (another 0.6 drop in TER). By encouraging the search along other decoder's decoding paths, we observed additional 0.15 BLEU improvement and 0.2 TER reduction. Finally, sentence translation hypothesis selection with word-based LM led to 0.2 BLEU point improvement and 0.16 point reduction in TER. To summarize, with the hierarchical system combination framework, we achieved 2.4 BLEU point improvement over the best baseline system, and reduce the TER by 1.4 point.

Table 4 shows the system combination results on Chinese-English newswire translation. The test data is NIST MT03 Chinese-English evaluation test set. In addition to the 4 baseline MT systems, we also add another phrase-based MT system (Lee et al., 2006). The system combination improves over the best baseline system by 2 BLEU points, and reduce the TER score by 1.6 percent. Thanks to the long range constituent reordering capability of different baseline systems, the path imitation improves the BLEU score by 0.4 point.

We consistently notice improved translation quality with system combination on unstructured text and speech translations, as shown in Table 5 and 6. With one reference translation, we notice 1.2 BLEU point improvement over the baseline block decoder (with 2.5 point TER reduction) on web log translation and about 2.1 point BLEU improvement (with 0.9 point TER reduction) on Broadcast News speech translation.

[3] The same LM is also used during first pass decoding by both the block and the DTM decoders.

	BLEUr1n4c	TER
sys1	0.2011	61.46
sys2	0.2211	66.32
sys3	0.2074	61.21
sys4	0.1258	85.45
syscom	0.2221	60.54

Table 6: System combination results for Arabic-English speech translation.

5 Related Work

Many system combination research have been done recently. (Matusov et al., 2006) computes consensus translation by voting on a confusion network, which is created by pairwise word alignment of multiple baseline MT hypotheses. This is similar to the sentence- and word- level combinations in (Rosti et al., 2007), where TER is used to align multiple hypotheses. Both approaches adopt black-box combination strategy, as target translations are combined independent of source sentences. (Rosti et al., 2007) extracts phrase translation pairs in the phrase level combination. Our proposed method incorporates bilingual information from source and target sentences in a hierarchical framework: word, phrase and decoding path combinations. Such information proves very helpful in our experiments. We also developed a path matching cost function to encourage decoding path imitation, thus enable one decoder to take advantage of rich reordering models of other MT systems. We only combine top-one hypothesis from each system, and did not apply system confidence measure and minimum error rate training to tune system combination weights. This will be our future work.

6 Conclusion

Our hierarchical system combination strategy effectively integrates word and phrase translation combinations, decoding path imitation and sentence hypothesis selection from multiple MT systems. By boosting common word and phrase translation pairs and pruning unused ones, we obtain better translation quality with less re-decoding time. By imitating the decoding paths, we take advantage of various reordering schemes from different decoders. The sentence hypothesis selection based on N-gram language model further improves the translation quality. The effectiveness has been consistently proved in several empirical studies with test sets in different languages and covering different genres.

7 Acknowledgment

The authors would like to thank Yaser Al-Onaizan, Abraham Ittycheriah and Salim Roukos for helpful discussions and suggestions. This work is supported under the DARPA GALE project, contract No. HR0011-06-2-0001.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion Models for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 529–536, Sydney, Australia, July. Association for Computational Linguistics.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The Mathematic of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Ahmad Emami, Kishore Papineni, and Jeffrey Sorensen. 2007. Large-scale Distributed Language Modeling. In *Proceedings of the 2007 International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2007)*, Honolulu, Hawaii, April.

Sanjika Hewavitharana, Bing Zhao, Almut Silja Hildebrand, Matthias Eck, Chiori Hori, Stephan Vogel, and Alex Waibel. 2005. The CMU Statistical Machine Translation System for IWSLT2005. In *Proceedings of IWSLT 2005*, Pittsburgh, PA, USA, November.

Arraham Ittycheriah and Salim Roukos. 2007. Direct Translation Model2. In *Proceedings of the 2007 Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007)*, Rochester, NY, April. Association for Computational Linguistics.

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-Engine Machine Translation Guided by Explicit Word Matching. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 101–104, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Y-S. Lee, S. Roukos, Y. Al-Onaizan, and K. Papineni. 2006. IBM Spoken Language Translation System. In *Proc. of TC-STAR Workshop on Speech-to-Speech Translation*, Barcelona, Spain.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing Consensus Translation for Multiple Machine Translation Systems Using Enhanced Hypothesis Alignment. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL '06)*, pages 263–270, Trento, Italy, April. Association for Computational Linguistics.

B. Mellebeek, K. Owczarzak, J. Van Genabith, and A. Way. 2006. Multi-Engine Machine Translation by Recursive Sentence Decomposition. In *Proceedings of the 7th biennial conference of the Association for Machine Translation in the Americas*, pages 110–118, Boston, MA, June.

Sergei Nirenburg and Robert Frederking. 1994. Toward Multi-engine Machine Translation. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 147–151, Morristown, NJ, USA. Association for Computational Linguistics.

Tadashi Nomoto. 2004. Multi-Engine Machine Translation with Voted Language Model. In *Proceedings of ACL*, pages 494–501.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie J. Dorr. 2007. Combining Translations from Multiple Machine Translation Systems. In *Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL'2007)*, Rochester, NY, April.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*.

D. Tidhar and U. Kussner. 2000. Learning to Select a Good Translation. In *Proceedings of the International Conference on Computational Linguistics*, pages 843–849.

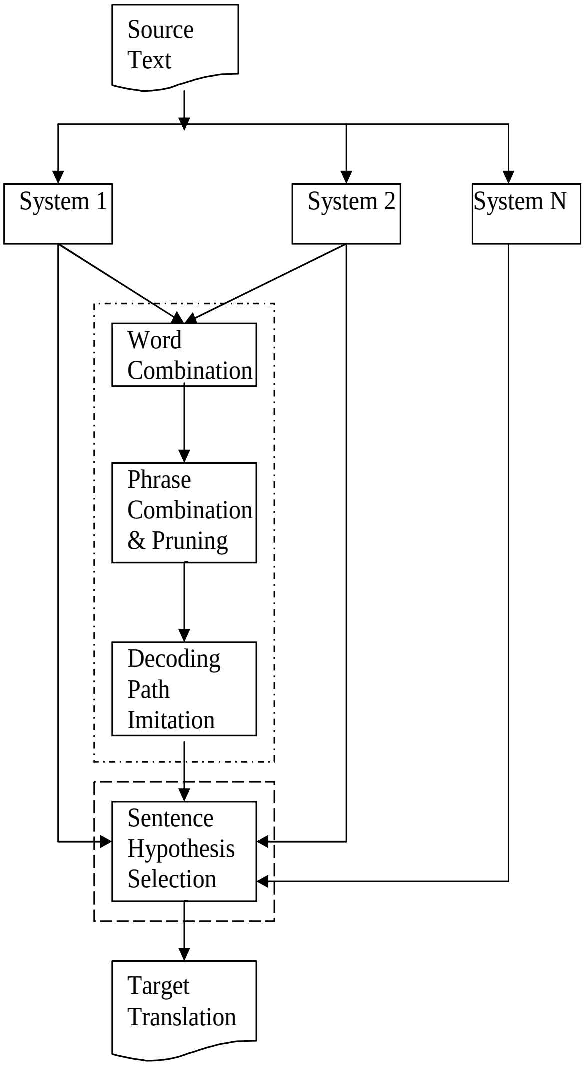


Figure 1: Hierarchical MT system combination architecture. The top dot-line rectangle is similar to the glass-box combination, and the bottom rectangle with sentence selection is similar to the black-box combination.

Original Sentence:

in *short* , making a good plan at the *beginning* of the construction is the *crucial measure* for *reducing haphazard* economic development .

Word-POS mixed stream:

in JJ , making a good plan at the NN of the construction is the JJ NN for VBG JJ economic development .

Figure 3: Sentence with Word-POS mixed stream.

Using RBMT Systems to Produce Bilingual Corpus for SMT

Xiaoguang Hu, Haifeng Wang, Hua Wu
Toshiba (China) Research and Development Center
5/F., Tower W2, Oriental Plaza
No.1, East Chang An Ave., Dong Cheng District
Beijing, 100738, China
{huxiaoguang, wanghaifeng, wuhua}@rdc.toshiba.com.cn

Abstract

This paper proposes a method using the existing Rule-based Machine Translation (RBMT) system as a black box to produce synthetic bilingual corpus, which will be used as training data for the Statistical Machine Translation (SMT) system. We use the existing RBMT system to translate the monolingual corpus into synthetic bilingual corpus. With the synthetic bilingual corpus, we can build an SMT system even if there is no real bilingual corpus. In our experiments using BLEU as a metric, the system achieves a relative improvement of 11.7% over the best RBMT system that is used to produce the synthetic bilingual corpora. We also interpolate the model trained on a real bilingual corpus and the models trained on the synthetic bilingual corpora. The interpolated model achieves an absolute improvement of 0.0245 BLEU score (13.1% relative) as compared with the individual model trained on the real bilingual corpus.

1 Introduction

Within the Machine Translation (MT) field, by far the most dominant paradigm is SMT, but many existing commercial systems are rule-based. In this research, we are interested in answering the question of whether the existing RBMT systems could be helpful to the development of an SMT system. To find the answer, let us first consider the following facts:

- Existing RBMT systems are usually provided as a black box. To make use of such systems, the most convenient way might be working on the translation results directly.
- SMT methods rely on bilingual corpus. As a data driven method, SMT usually needs large bilingual corpus as the training data.

Based on the above facts, in this paper we propose a method using the existing RBMT system as a black box to produce a synthetic bilingual corpus[1], which will be used as the training data for the SMT system.

For a given language pair, the monolingual corpus is usually much larger than the real bilingual corpus. We use the existing RBMT system to translate the monolingual corpus into synthetic bilingual corpus. Then, even if there is no real bilingual corpus, we can train an SMT system with the monolingual corpus and the synthetic bilingual corpus. If there exist *n* available RBMT systems for the desired language pair, we use the *n* systems to produce *n* synthetic bilingual corpora, and *n* translation models are trained with the *n* corpora respectively. We name such a model the *synthetic model*. An interpolated translation model is built by linear interpolating the *n* synthetic models. In our experiments using BLEU (Papineni et al., 2002) as the metric, the interpolated synthetic model achieves a relative improvement of 11.7% over the best RBMT system that is used to produce the synthetic bilingual corpora.

[1] In this paper, to be distinguished from the real bilingual corpus, the bilingual corpus generated by the RBMT system is called a synthetic bilingual corpus.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 287–295, Prague, June 2007.

Moreover, if a real bilingual corpus is available for the desired language pair, we build another translation model, which is named the *standard model*. Then we can build an *interpolated model* by interpolating the standard model and the synthetic models. Experimental results show that the interpolated model achieves an absolute improvement of 0.0245 BLEU score (13.1% relative) as compared with the standard model.

The remainder of this paper is organized as follows. In section 2 we summarize the related work. We then describe our method Using RBMT systems to produce bilingual corpus for SMT in section 3. Section 4 describes the resources used in the experiments. Section 5 presents the experiment result, followed by the discussion in section 6. Finally, we conclude and present the future work in section 7.

2 Related Work

In the MT field, by far the most dominant paradigm is SMT. SMT has evolved from the original word-based approach (Brown et al., 1993) into phrase-based approaches (Koehn et al., 2003; Och and Ney, 2004) and syntax-based approaches (Wu, 1997; Alshawi et al., 2000; Yamada and Knignt, 2001; Chiang, 2005). On the other hand, much important work continues to be carried out in Example-Based Machine Translation (EBMT) (Carl et al., 2005; Way and Gough, 2005), and many existing commercial systems are rule-based.

Although we are not aware of any previous attempt to use an existing RBMT system as a black box to produce synthetic bilingual training corpus for general purpose SMT systems, there exists a great deal of work on MT hybrids and Multi-Engine Machine Translation (MEMT).

Research into MT hybrids has increased over the last few years. Some research focused on the hybrid of various corpus-based MT methods, such as SMT and EBMT (Vogel and Ney, 2000; Marcu, 2001; Groves and Way, 2006; Menezes and Quirk, 2005). Others tried to exploit the advantages of both rule-based and corpus-based methods. Habash et al. (2006) built an Arabic-English generation-heavy MT system and boosted it with SMT components. METIS-II is a hybrid machine translation system, in which insights from SMT, EBMT, and RBMT are used (Vandeghinste et al., 2006). Seneff et al. (2006) combined an interlingual translation framework with phrase-based SMT for spoken language translation in a limited domain. They automatically generated a corpus of English-Chinese pairs from the same interlingual representation by parsing the English corpus and then paraphrasing each utterance into both English and Chinese.

Frederking and Nirenburg (1994) produced the first MEMT system by combining outputs from three different MT engines based on their knowledge of the inner workings of the engines. Nomoto (2004) used voted language models to select the best output string at sentence level. Some recent approaches to MEMT used word alignment techniques for comparison between the MT systems (Jayaraman and Lavie, 2005; Zaanen and Somers, 2005; Matusov et al. 2006). All the above MEMT systems operate on MT outputs for complete input sentences. Mellebeek et al. (2006) presented a different approach, using a recursive decomposition algorithm that produces simple chunks as input to the MT engines. A consensus translation is produced by combining the best chunk translation.

This paper uses RBMT outputs to improve the performance of SMT systems. Instead of RBMT outputs, other researchers have used SMT outputs to boost translation quality. Callision-Burch and Osborne (2003) used co-training to extend existing parallel corpora, wherein machine translations are selectively added to training corpora with multiple source texts. They also created training data for a language pair without a parallel corpus by using multiple source texts. Ueffing (2006) explored monolingual source-language data to improve an existing machine translation system via self-training. The source data is translated by a SMT system, and the reliable translations are automatically identified. Both of the methods improved translation quality.

3 Method

In this paper, we use the synthetic and real bilingual corpus to train the phrase-based translation models.

3.1 Phrase-Based Models

According to the translation model presented in (Koehn et al., 2003), given a source sentence f, the best target translation e_{best} can be obtained using the following model

$$\begin{aligned} \mathrm{e}_{\mathrm{best}} &= \arg\max_{\mathrm{e}} p(\mathrm{e} \mid \mathrm{f}) \\ &= \arg\max_{\mathrm{e}} p(\mathrm{f} \mid \mathrm{e}) p_{\mathrm{LM}}(\mathrm{e}) \omega^{\mathrm{length(e)}} \end{aligned} \quad (1)$$

Where the translation model $p(\mathrm{f} \mid \mathrm{e})$ can be decomposed into

$$\begin{aligned} & p(\overline{f}_1^I \mid \overline{e}_1^I) \\ &= \prod_{i=1}^{I} \phi(\overline{f}_i \mid \overline{e}_i) d(a_i - b_{i-1}) p_{\mathrm{w}}(\overline{f}_i \mid \overline{e}_i, a)^{\lambda} \end{aligned} \quad (2)$$

Where $\phi(\overline{f}_i \mid \overline{e}_i)$ is the phrase translation probability. a_i denotes the start position of the source phrase that was translated into the *i*th target phrase, and b_{i-1} denotes the end position of the source phrase translated into the (*i*-1)th target phrase. $d(a_i - b_{i-1})$ is the distortion probability. $p_{\mathrm{w}}(\overline{f}_i \mid \overline{e}_i, a)$ is the lexical weight, and λ is the strength of the lexical weight.

3.2 Interpolated Models

We train synthetic models with the synthetic bilingual corpus produced by the RBMT systems. We can also train a translation model, namely standard model, if a real bilingual corpus is available. In order to make full use of these two kinds of corpora, we conduct linear interpolation between them.

In this paper, the distortion probability in equation (2) is estimated during decoding, using the same method as described in Pharaoh (Koehn, 2004). For the phrase translation probability and lexical weight, we interpolate them as shown in (3) and (4).

$$\phi(\overline{f} \mid \overline{e}) = \sum_{i=0}^{n} \alpha_i \phi_i(\overline{f} \mid \overline{e}) \quad (3)$$

$$p_{\mathrm{w}}(\overline{f} \mid \overline{e}, a) = \sum_{i=0}^{n} \beta_i p_{\mathrm{w},i}(\overline{f} \mid \overline{e}, a) \quad (4)$$

Where $\phi_0(\overline{f} \mid \overline{e})$ and $p_{\mathrm{w},0}(\overline{f} \mid \overline{e}, a)$ denote the phrase translation probability and lexical weight trained with the real bilingual corpus, respectively. $\phi_i(\overline{f} \mid \overline{e})$ and $p_{\mathrm{w},i}(\overline{f} \mid \overline{e}, a)$ ($i = 1,...,n$) are the phrase translation probability and lexical weight estimated by *n* synthetic corpora produced by the RBMT systems. α_i and β_i are interpolation coefficients, ensuring $\sum_{i=0}^{n} \alpha_i = 1$ and $\sum_{i=0}^{n} \beta_i = 1$.

4 Resources Used in Experiments

4.1 Data

In the experiments, we take English-Chinese translation as a case study. The real bilingual corpus includes 494,149 English-Chinese bilingual sentence pairs. The monolingual English corpus is selected from the English Gigaword Second Edition, which is provided by Linguistic Data Consortium (LDC) (catalog number LDC2005T12). The selected monolingual corpus includes 1,087,651 sentences.

For language model training, we use part of the Chinese Gigaword Second Edition provided by LDC (catalog number LDC2005T14). We use 41,418 documents selected from the ZaoBao Newspaper and 992,261 documents from the XinHua News Agency to train the Chinese language model, amounting to 5,398,616 sentences.

The test set and the development set are from the corpora distributed for the 2005 HTRDP [2] evaluation of machine translation. It can be obtained from Chinese Linguistic Data Consortium (catalog number 2005-863-001). We use the same 494 sentences in the test set and 278 sentences in the development set. Each source sentence in the test set and the development set has 4 different references.

4.2 Tools

In this paper, we use two off-the-shelf commercial English to Chinese RBMT systems to produce the synthetic bilingual corpus.

We also need a trainer and a decoder to perform phrase-based SMT. We use Koehn's training scripts [3] to train the translation model, and the SRILM toolkit (Stolcke, 2002) to train language model. For the decoder, we use Pharaoh (Koehn, 2004). We run the decoder with its default settings (maximum phrase length 7) and then use Koehn's implementation of minimum error rate training (Och, 2003) to tune the feature weights on the de-

[2] The full name of HTRDP is National High Technology Research and Development Program of China, also named as 863 Program.

[3] It is located at http://www.statmt.org/wmt06/shared-task/baseline.html.

velopment set. The translation quality is evaluated using a well-established automatic measure: BLEU score (Papineni et al., 2002). We use the same method described in (Koehn and Monz, 2006) to perform the significance test.

5 Experimental Results

5.1 Results on Synthetic Corpus Only

With the monolingual English corpus and the English side of the real bilingual corpus, we translate them into Chinese using the two commercial RBMT systems and produce two synthetic bilingual corpora. With the corpora, we train two synthetic models as described in section 3.1. Based on the synthetic models, we also perform linear interpolation as shown in section 3.2, without the standard models. We tune the interpolation weights using the development set, and achieve the best performance when $\alpha_1 = 0.58$, $\alpha_2 = 0.42$, $\beta_1 = 0.58$, and $\beta_2 = 0.42$. The translation results on the test set are shown in Table 1. Synthetic model 1 and 2 are trained using the synthetic bilingual corpora produced by RBMT system 1 and RBMT system 2, respectively.

Method	BLEU
RBMT system 1	0.1681
RBMT system 2	0.1453
Synthetic Model 1	0.1644
Synthetic Model 2	0.1668
Interpolated Synthetic Model	0.1878

Table 1. Translation Results Using Synthetic Bilingual Corpus

From the results, it can be seen that the interpolated synthetic model obtains the best result, with an absolute improvement of the 0.0197 BLEU (11.7% relative) as compared with RBMT system 1, and 0.0425 BLEU (29.2% relative) as compared with RBMT system 2. It is very promising that our method can build an SMT system that significantly outperforms both of the two RBMT systems, using the synthetic bilingual corpus produced by two RBMT systems.

5.2 Results on Real and Synthetic Corpus

With the real bilingual corpus, we build a standard model. We interpolate the standard model with the two synthetic models built in section 5.1 to obtain interpolated models. The translation results are shown in Table 2. The interpolation coefficients are both for phrase table probabilities and lexical weights. They are also tuned using the development set.

From the results, it can be seen that all the three interpolated models perform not only better than the RBMT systems but also better than the SMT system trained on the real bilingual corpus. The interpolated model combining the standard model and the two synthetic models performs the best, achieving a statistically significant improvement of about 0.0245 BLEU (13.1% relative) as compared with the standard model with no synthetic corpus. It also achieves 26.1% and 45.8% relative improvement as compared with the two RBMT systems respectively. The results indicate that using the corpus produced by RBMT systems, the performance of the SMT system can be greatly improved. The results also indicate that the more the RBMT systems are used, the better the translation quality is.

Interpolation Coefficients			BLEU
Standard model	Synthetic Model 1	Synthetic Model 2	
1	—	—	0.1874
0.90	0.10	—	0.2056
0.86	—	0.14	0.2040
0.70	0.12	0.18	0.2119

Table 2. Translation Results Using Standard and Synthetic Bilingual Corpus

5.3 Effect of Synthetic Corpus Size

To explore the relationship between the translation quality and the scale of the synthetic bilingual corpus, we interpolate the standard model with the synthetic models trained with synthetic bilingual corpus of different sizes. In order to simplify the procedure, we only use RBMT system 1 to translate the 1,087,651 monolingual English sentences to produce the synthetic bilingual corpus.

We randomly select 20%, 40%, 60%, 80%, and 100% of the synthetic bilingual corpus to train different synthetic models. The translation results of the interpolated models are shown in Figure 1. The results indicate that the larger the synthetic bilingual corpus is, the better translation performance would be.

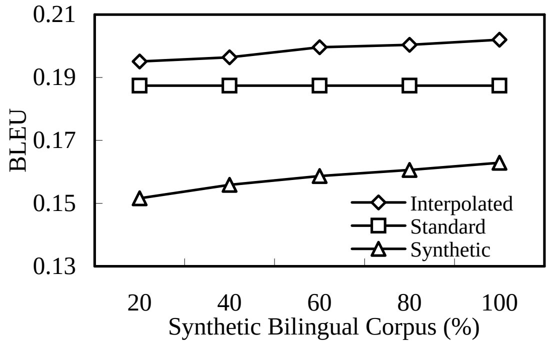


Figure 1. Comparison of Translation Results Using Synthetic Bilingual Corpus of Different Sizes

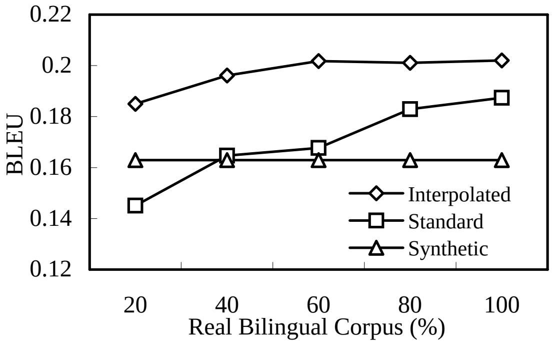


Figure 2. Comparison of Translation Results Using Real Bilingual Corpus of Different Sizes

5.4 Effect of Real Corpus Size

Another issue is the relationship between the SMT performance and the size of the real bilingual corpus. To train different standard models, we randomly build five corpora of different sizes, which contain 20%, 40%, 60%, 80%, and 100% sentence pairs of the real bilingual corpus, respectively. As to the synthetic model, we use the same synthetic model 1 that is described in section 5.1. Then we build five interpolated models by performing linear interpolation between the synthetic model and the five standard models respectively. The translation results are shown in Figure 2.

From the results, we can see that the larger the real bilingual corpus is, the better the performance of both standard models and interpolated models would be. The relative improvement of BLEU scores is up to 27.5% as compared with the corresponding standard models.

5.5 Results without Additional Monolingual Corpus

In all the above experiments, we use an additional English monolingual corpus to get more synthetic bilingual corpus. We are also interested in the results without the additional monolingual corpus. In such case, the only English monolingual corpus is the English side of the real bilingual corpus. We use this smaller size of monolingual corpus and the real bilingual corpus to conduct similar experiments as in section 5.2. The translation results are shown in Table 3.

From the results, it can be seen that our method works well even if no additional monolingual corpus is available. We achieve a statistically significant improvement of about 0.01 BLEU (5.2% relative) as compared with the standard model without using the synthetic corpus.

Interpolation Coefficients			BLEU
Standard model	Synthetic Model 1	Synthetic Model 2	
1	—	—	0.1874
—	1	—	0.1560
—	—	1	0.1522
0.80	0.10	0.10	0.1972

Table 3. Translation Results without Additional Monolingual Corpus

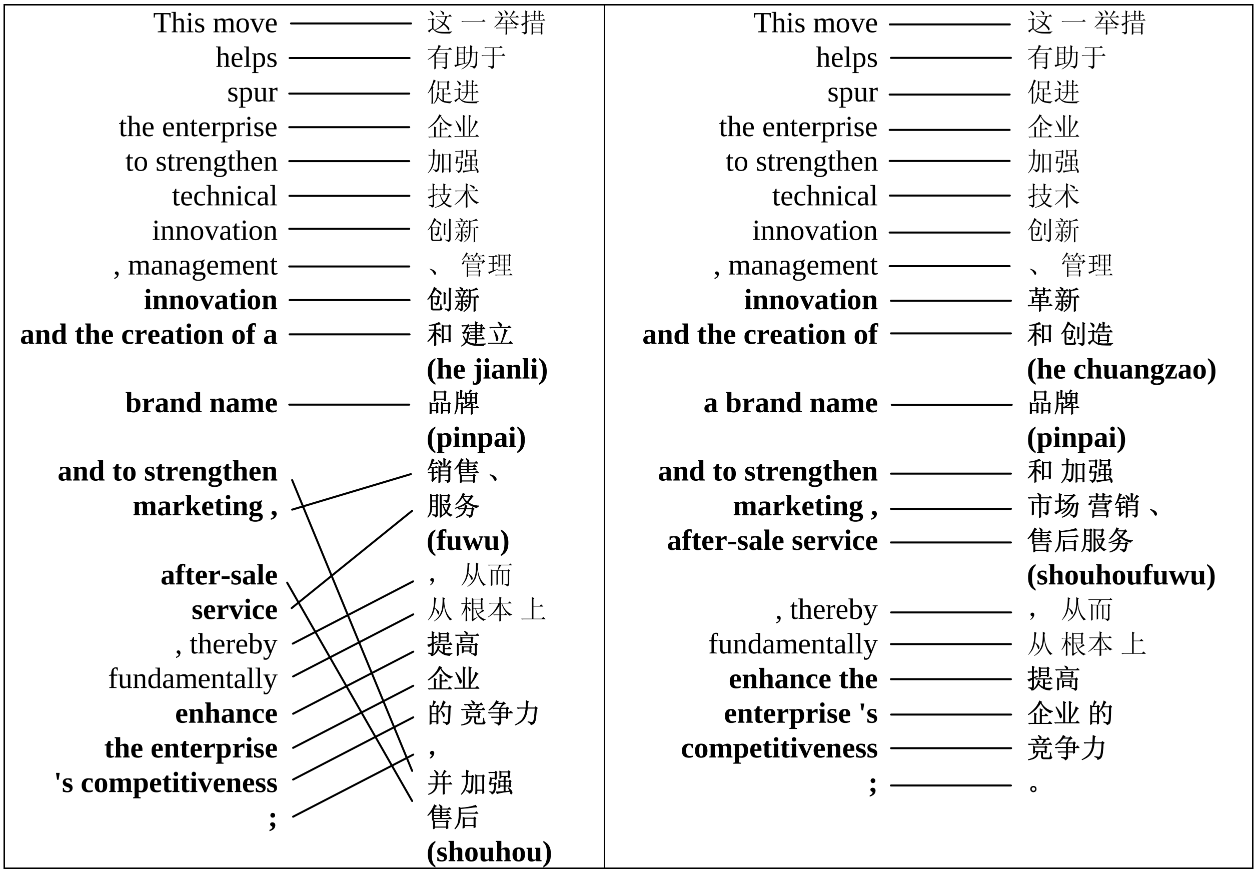
	Standard Model	Synthetic Model 1	Synthetic Model 2
Standard Model	6,105,260	—	—
Synthetic Model 1	356,795	12,062,068	—
Synthetic Model 2	357,489	881,921	9,216,760

Table 4. Numbers of Phrase Pairs

In order to further analyze the translation results, we examine the overlap and the difference among the phrase tables. The analytic results are shown in Table 4. More phrase pairs are extracted by the synthetic models, about twice by the synthetic model 1 in particular, than those extracted by the standard model. The overlap between each model is very low. For example, about 6% phrase pairs extracted by the standard model make appearance in both the standard model and the synthetic model 1. This also explains why the interpolated model outperforms that of the standard model in Table 3.

Methods	English Sentence / Chinese Translations	BLEU
	This move helps spur the enterprise to strengthen technical innovation, management innovation and the creation of a brand name and to strengthen marketing, after-sale service, thereby fundamentally enhance the enterprise's competitiveness;	
Standard model	这 一 举措 有助于 促进 企业 加强 技术 创新 、 管理 创新 和 建立 品牌 销售 、 服务 ， 从而 从 根本 上 提高 企业 的 竞争力 ， 并 加强 售后	0.5022
RBMT System 1	这种 行动 帮助 刺激 企业 加强 技术 地 革新 ， 管理 革新 和 创造 一个 名牌 并且 加强 销售 ， 在 销售 服务 ， 基本上 进而 提高 企业 的 竞争 。	0.1535
RBMT System 2	这项 行动 帮助 刺激 这家 企业 加强 技术 发明 、 管理 创新 和 一 个 商标 的 创造 并 加强 市场 销售 ， 因此 售后服务 根本 增强 这家 企业 的 竞争 。	0.1485
Interpolated Model	这 一 举措 有助于 促进 企业 加强 技术 创新 、 管理 革新 和 创造 品牌 和 加强 市场 营销 、 售后服务 ， 从而 从 根本 上 提高 企业 的 竞争力 。	0.7198

Table 5. Translation Example



(a) Results Produced by the Standard Model (b) Results Produced by the Interpolated Model

Figure 3. Phrase Pairs Used for Translation

6 Discussion

6.1 Model Interpolation vs. Corpus Merge

In section 5, we make use of the real bilingual corpus and the synthetic bilingual corpora by performing model interpolation. Another available way is directly combining these two kinds of corpora to train a translation model, namely corpus merge. In order to compare these two methods, we use RBMT system 1 to translate the 1,087,651 monolingual English sentences to produce synthetic bilingual corpus. Then we train an SMT system with the combination of this synthetic bilingual corpus and the real bilingual corpus. The BLEU score of such system is 0.1887, while that of the model interpolation system is 0.2020. It indicates that the model interpolation method is significantly better than the corpus merge method.

6.2 Result Analysis

As discussed in Section 5.5, the number of the overlapped phrase pairs among the standard model and the synthetic models is very small. The newly added phrase pairs from the synthetic models can assist to improve the translation results of the interpolated model. In this section, we will use an example to further discuss the reason behind the improvement of the SMT system by using synthetic bilingual corpus. Table 5 shows an English sentence and its Chinese translations produced by different methods. And Figure 3 shows the phrase pairs used for translation. The results show that imperfect translations of RBMT systems can be also used to boost the performance of an SMT system.

	Phrase Pairs	Phrase Pairs Used	New Pairs Used
Standard Model	6,105,260	5,509	—
Interpolated Model	73,221,525	5,306	1993

Table 6. Statistics of Phrase Pairs

Further analysis is shown in Table 6. After adding the synthetic corpus produced by the RBMT systems, the interpolated model outperforms the standard models mainly for the following two reasons: (1) some new phrase pairs are added into the interpolated model. 37.6% phrase pairs (1993 out of 5306) are newly learned and used for translation. For example, the phrase pair "after-sale service <-> 售后服务 (shouhoufuwu)" is added; (2) The probability distribution of the phrase pairs is changed. For example, the probabilities of the two pairs "a brand name <-> 品牌 (pinpai)" and "and the creation of <-> 和 创造 (he chuangzao)" increase. The probabilities of the other two pairs "brand name <-> 品牌 (pinpai)" and "and the creation of a <-> 和 建立 (he jianli)" decrease. We found that 930 phrase pairs, which are also in the phrase table of the standard model, are used by the interpolated model for translation but not used by the standard model.

6.3 Human Evaluation

According to (Koehn and Monz, 2006; Callison-Burch et al., 2006), the RBMT systems are usually not adequately appreciated by BLEU. We also manually evaluated the RBMT systems and SMT systems in terms of both adequacy and fluency as defined in (Koehn and Monz, 2006). The evaluation results show that the SMT system with the interpolated model, which achieves the highest BLEU scores in Table 2, achieves slightly better adequacy and fluency scores than the two RBMT systems.

7 Conclusion and Future Work

We presented a method using the existing RBMT system as a black box to produce synthetic bilingual corpus, which was used as training data for the SMT system. We used the existing RBMT system to translate the monolingual corpus into a synthetic bilingual corpus. With the synthetic bilingual corpus, we could build an SMT system even if there is no real bilingual corpus. In our experiments using BLEU as the metric, such a system achieves a relative improvement of 11.7% over the best RBMT system that is used to produce the synthetic bilingual corpora. It indicates that using the existing RBMT systems to produce a synthetic bilingual corpus, we can build an SMT system that outperforms the existing RBMT systems.

We also interpolated the model trained on a real bilingual corpus and the models trained on the synthetic bilingual corpora, the interpolated model achieves an absolute improvement of 0.0245 BLEU score (13.1% relative) as compared with the individual model trained on the real bilingual corpus. It indicates that we can build a better SMT system by leveraging the real and the synthetic bilingual corpus.

Further result analysis shows that after adding the synthetic corpus produced by the RBMT systems, the interpolated model outperforms the standard models mainly because of two reasons: (1) some new phrase pairs are added to the interpolated model; (2) the probability distribution of the phrase pairs is changed.

In the future work, we will investigate the possibility of training a reverse SMT system with the RBMT systems. For example, we will investigate to train Chinese-to-English SMT system based on natural English and RBMT-generated synthetic Chinese.

References

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning Dependency Translation Models as Collections of Finite-State Head Transducers. *Computational Linguistics*, 26(1): 45-60.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2): 263-311.

Chris Callison-Burch and Miles Osborne. 2003. Bootstrapping Parallel Corpora. In *Proceedings of the Human Language Technology conference / North American chapter of the Association for Computational Linguistics (HLT/NAACL-2003) Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 44-49.

Chris Callison-Burch, Miles Osborne and Philipp Koehn, 2006. Re-evaluating the Role of Bleu in Machine Translation Research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 249-256.

Michel Carl, Paul Schmidt, and Jorg Schutz. 2005. Reversible Template-based Shake & Bake Generation. In *Proceedings of the 10th Machine Translation Summit Workshop on Example-Based Machine Translation*, pages 17-25.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, pages 263-270.

Robert Frederking and Sergei Nirenburg. 1994. Three Heads Are Better Than One. In *Proceedings of the 4th Applied Natural Language Processing Conference (ANLP-1994)*, pages 95-100.

Declan Groves and Andy Way. 2006. Hybridity in MT: Experiments on the Europarl Corpus. In *Proceedings of the 11th Annual Conference of the European Association for Machine Translation (EAMT-2006)*, pages 115-124.

Nizar Habash, Bonnie Dorr, and Christof Monz. 2006 Challenges in Building an Arabic-English GHMT System with SMT Components. In *Proceedings of the 11th Annual Conference of the European Association for Machine Translation (EAMT-2006)*, pages 56-65.

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-engine Machine Translation Guided by Explicit Word Matching. In *Proceedings of the 10th Annual Conference of the European Association for Machine Translation (EAMT-2005)*, pages 143-152.

Philipp Koehn. 2004. Pharaoh: A Beam Search Decoder For Phrase-Based Statistical Machine Translation Models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA-2004)*, pages 115-124.

Philipp Koehn and Christof Monz. 2006. Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings of the Human Language Technology conference / North American Chapter of the Association for Computational Linguistics (HLT/NAACL-2006) Workshop on Statistical Machine Translation*, pages 102-121.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the Human Language Technology conference / North American Chapter of the Association for Computational Linguistics (HLT/NAACL-2003)*, pages 127-133.

Daniel Marcu. 2001. Towards a Unified Approach to Memory- and Statistical-based Machine Translation. In *Proceedings of the Association for Computational Linguistics / European Chapter of the Association for Computational Linguistics (ACL/EACL-2001)*, pages 378-385.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 33-40.

Bart Mellebeek, Karolina Owczarzak, Josef Van Genabith, and Andy Way. 2006. Multi-engine Machine Translation by Recursive Sentence Decomposition. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-2006)*, pages 110-118.

Arul Menezes and Chris Quirk. 2005. Dependency treelet translation: the convergence of statistical and example-based machine-translation? In *Proceedings of the 10th Machine Translation Summit Workshop on Example-Based Machine Translation*, pages 99-108.

Tadashi Nomoto. 2004. Multi-Engine machine translation with voted language model. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pages 494-501.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pages 160-167.

Franz Josef Och and Hermann Ney. 2004. The Alignment Template Approach To Statistical Machine Translation. *Computational Linguistics*, 30(4):417-449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 311-318.

Stephanie Seneff, Chao Wang, and John Lee. 2006. Combining Linguistic and Statistical Methods for Bi-Directional English Chinese Translation in the Flight Domain. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-2006)*, pages 213-222.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP-2002)*, pages 901-904.

Nicola Ueffing. 2006. Using Monolingual Source-Language Data to Improve MT Performance. *In Proceedings of the International Workshop on Spoken Language Translation (IWSLT-2006)*, pages 174-181.

Vincent Vandeghinste, Ineka Schuurman, Michael Carl, Stella Markantonatou, and Toni Badia. 2006. Metis-II: Machine Translation for Low-Resource Languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (L-REC-2006)*, pages 1284-1289.

Stephan Vogel and Hermann Ney. 2000. Construction of a Hierarchical Translation Memory. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 1131-1135.

Andy Way and Nano Gough. 2005. Comparing Example-Based and Statistical Machine Translation. *Natural Language Engineering*, 11(3): 295-309.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3): 377-403.

Kenji Yamada and Kevin Knight. 2001. A Syntax Based Statistical Translation Model. In *Proceedings of the Association for Computational Linguistics / European Chapter of the Association for Computational Linguistics (ACL/EACL-2001)*, pages 523-530.

Menno van Zaanen and Harold Somers. 2005. DEMOCRAT: Deciding between Multiple Outputs Created by Automatic Translation. In *Proceedings of the 10th Machine Translation Summit*, pages 173-180.

Why doesn't EM find good HMM POS-taggers?

Mark Johnson

Microsoft Research
Redmond, WA
t-majoh@microsoft.com

Brown University
Providence, RI
Mark_Johnson@Brown.edu

Abstract

This paper investigates why the HMMs estimated by Expectation-Maximization (EM) produce such poor results as Part-of-Speech (POS) taggers. We find that the HMMs estimated by EM generally assign a roughly equal number of word tokens to each hidden state, while the empirical distribution of tokens to POS tags is highly skewed. This motivates a Bayesian approach using a sparse prior to bias the estimator toward such a skewed distribution. We investigate Gibbs Sampling (GS) and Variational Bayes (VB) estimators and show that VB converges faster than GS for this task and that VB significantly improves 1-to-1 tagging accuracy over EM. We also show that EM does nearly as well as VB when the number of hidden HMM states is dramatically reduced. We also point out the high variance in all of these estimators, and that they require many more iterations to approach convergence than usually thought.

1 Introduction

It is well known that Expectation-Maximization (EM) performs poorly in unsupervised induction of linguistic structure (Carroll and Charniak, 1992; Merialdo, 1994; Klein, 2005; Smith, 2006). In retrospect one can certainly find reasons to explain this failure: after all, likelihood does not appear in the wide variety of linguistic tests proposed for identifying linguistic structure (Fromkin, 2001).

This paper focuses on unsupervised part-of-speech (POS) tagging, because it is perhaps the simplest linguistic induction task. We suggest that one reason for the apparent failure of EM for POS tagging is that it tends to assign relatively equal numbers of tokens to each hidden state, while the empirical distribution of POS tags is highly skewed, like many linguistic (and non-linguistic) phenomena (Mitzenmacher, 2003). We focus on first-order Hidden Markov Models (HMMs) in which the hidden state is interpreted as a POS tag, also known as bitag models.

In this setting we show that EM performs poorly when evaluated using a "1-to-1 accuracy" evaluation, where each POS tag corresponds to at most one hidden state, but is more competitive when evaluated using a "many-to-1 accuracy" evaluation, where several hidden states may correspond to the same POS tag. We explain this by observing that the distribution of hidden states to words proposed by the EM-estimated HMMs is relatively uniform, while the empirical distribution of POS tags is heavily skewed towards a few high-frequency tags. Based on this, we propose a Bayesian prior that biases the system toward more skewed distributions and show that this raises the 1-to-1 accuracy significantly. Finally, we show that a similar increase in accuracy can be achieved by reducing the number of hidden states in the models estimated by EM.

There is certainly much useful information that bitag HMMs models cannot capture. Toutanova et al. (2003) describe a wide variety of morphological and distributional features useful for POS tagging, and Clark (2003) proposes ways of incorporating some of these in an unsupervised tagging model. However, bitag models are rich enough to capture at least some distributional information (i.e., the tag

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 296–305, Prague, June 2007.

for a word depends on the tags assigned to its neighbours). Moreover, more complex models add additional complicating factors that interact in ways still poorly understood; for example, smoothing is generally regarded as essential for higher-order HMMs, yet it is not clear how to integrate smoothing into unsupervised estimation procedures (Goodman, 2001; Wang and Schuurmans, 2005).

Most previous work exploiting unsupervised training data for inferring POS tagging models has focused on semi-supervised methods in the in which the learner is provided with a lexicon specifying the possible tags for each word (Merialdo, 1994; Smith and Eisner, 2005; Goldwater and Griffiths, 2007) or a small number of "prototypes" for each POS (Haghighi and Klein, 2006). In the context of semi-supervised learning using a tag lexicon, Wang and Schuurmans (2005) observe discrepencies between the empirical and estimated tag frequencies similar to those observed here, and show that constraining the estimation procedure to preserve the empirical frequencies improves tagging accuracy. (This approach cannot be used in an unsupervised setting since the empirical tag distribution is not available). However, as Banko and Moore (2004) point out, the accuracy achieved by these unsupervised methods depends strongly on the precise nature of the supervised training data (in their case, the ambiguity of the tag lexicon available to the system), which makes it more difficult to understand the behaviour of such systems.

2 Evaluation

All of the experiments described below have the same basic structure: an estimator is used to infer a bitag HMM from the unsupervised training corpus (the words of Penn Treebank (PTB) Wall Street Journal corpus (Marcus et al., 1993)), and then the resulting model is used to label each word of that corpus with one of the HMM's hidden states. This section describes how we evaluate how well these sequences of hidden states correspond to the gold-standard POS tags for the training corpus (here, the PTB POS tags). The chief difficulty is determining the correspondence between the hidden states and the gold-standard POS tags.

Perhaps the most straightforward method of establishing this correspondence is to deterministically map each hidden state to the POS tag it co-occurs most frequently with, and return the proportion of the resulting POS tags that are the same as the POS tags of the gold-standard corpus. We call this the *many-to-1 accuracy* of the hidden state sequence because several hidden states may map to the same POS tag (and some POS tags may not be mapped to by any hidden states at all).

As Clark (2003) points out, many-to-1 accuracy has several defects. If a system is permitted to posit an unbounded number of hidden states (which is not the case here) then it can achieve a perfect many-to-1 accuracy by placing every word token into its own unique state. Cross-validation, i.e., identifying the many-to-1 mapping and evaluating on different subsets of the data, would answer many of these objections. Haghighi and Klein (2006) propose constraining the mapping from hidden states to POS tags so that at most one hidden state maps to any POS tag. This mapping is found by greedily assigning hidden states to POS tags until either the hidden states or POS tags are exhausted (note that if the number of hidden states and POS tags differ, some will be unassigned). We call the accuracy of the POS sequence obtained using this map its *1-to-1 accuracy*.

Finally, several authors have proposed using information-theoretic measures of the divergence between the hidden state and POS tag sequences. Goldwater and Griffiths (2007) propose using the *Variation of Information* (VI) metric described by Meilă (2003). We regard the assignments of hidden states and POS tags to the words of the corpus as two different ways of clustering those words, and evaluate the conditional entropy of each clustering conditioned on the other. The VI is the sum of these conditional entropies. Specifically, given a corpus labeled with hidden states and POS tags, if $\tilde{p}(y), \tilde{p}(t)$ and $\tilde{p}(y,t)$ are the empirical probabilities of a hidden state y, a POS tag t, and the cooccurance of y and t respectively, then the mutual information I, entropies H and variation of information VI are defined as follows:

$$\begin{aligned}
H(Y) &= -\sum_y \tilde{p}(y) \log \tilde{p}(y) \\
H(T) &= -\sum_t \tilde{p}(t) \log \tilde{p}(t) \\
I(Y,T) &= \sum_{y,t} \tilde{p}(y,t) \log \frac{\tilde{p}(y,t)}{\tilde{p}(y)\tilde{p}(t)} \\
H(Y|T) &= H(Y) - I(Y,T)
\end{aligned}$$

$$
\begin{aligned}
H(T|Y) &= H(T) - I(Y,T) \\
VI(Y,T) &= H(Y|T) + H(T|Y)
\end{aligned}
$$

As Meilă (2003) shows, VI is a metric on the space of probability distributions whose value reflects the divergence between the two distributions, and only takes the value zero when the two distributions are identical.

3 Maximum Likelihood via Expectation-Maximization

There are several excellent textbook presentations of Hidden Markov Models and the Forward-Backward algorithm for Expectation-Maximization (Jelinek, 1997; Manning and Schütze, 1999; Bishop, 2006), so we do not cover them in detail here. Conceptually, a Hidden Markov Model generates a sequence of observations $\mathbf{x} = (x_0, \ldots, x_n)$ (here, the words of the corpus) by first using a Markov model to generate a sequence of hidden states $\mathbf{y} = (y_0, \ldots, y_n)$ (which will be mapped to POS tags during evaluation as described above) and then generating each word x_i conditioned on its corresponding state y_i. We insert endmarkers at the beginning and ending of the corpus and between sentence boundaries, and constrain the estimator to associate endmarkers with a state that never appears with any other observation type (this means each sentence can be processed independently by first-order HMMs; these endmarkers are ignored during evaluation).

In more detail, the HMM is specified by multinomials θ_y and ϕ_y for each hidden state y, where θ_y specifies the distribution over states following y and ϕ_y specifies the distribution over observations x given state y.

$$
\begin{array}{lcl}
y_i \mid y_{i-1} = y & \sim & \mathrm{Multi}(\theta_y) \\
x_i \mid y_i = y & \sim & \mathrm{Multi}(\phi_y)
\end{array}
\tag{1}
$$

We used the Forward-Backward algorithm to perform Expectation-Maximization, which is a procedure that iteratively re-estimates the model parameters (θ, ϕ), converging on a local maximum of the likelihood. Specifically, if the parameter estimate at time ℓ is $(\theta^{(\ell)}, \phi^{(\ell)})$, then the re-estimated parameters at time $\ell + 1$ are:

$$
\begin{aligned}
\theta^{(\ell+1)}_{y'|y} &= \mathrm{E}[n_{y',y}]/\mathrm{E}[n_y] \qquad (2) \\
\phi^{(\ell+1)}_{x|y} &= \mathrm{E}[n_{x,y}]/\mathrm{E}[n_y]
\end{aligned}
$$

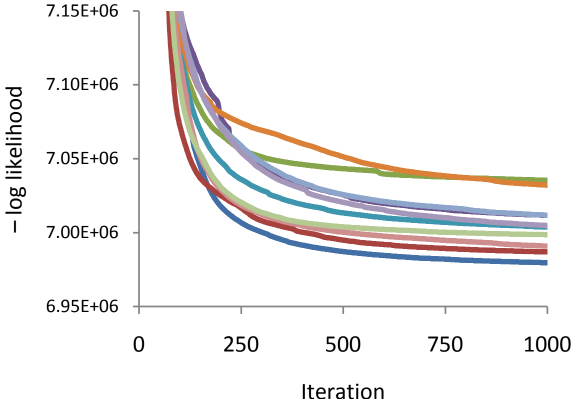


Figure 1: Variation in negative log likelihood with increasing iterations for 10 EM runs from different random starting points.

where $n_{x,y}$ is the number of times observation x occurs with state y, $n_{y',y}$ is the number of times state y' follows y and n_y is the number of occurences of state y; all expectations are taken with respect to the model $(\theta^{(\ell)}, \phi^{(\ell)})$.

We took care to implement this and the other algorithms used in this paper efficiently, since optimal performance was often only achieved after several hundred iterations. It is well-known that EM often takes a large number of iterations to converge in likelihood, and we found this here too, as shown in Figure 1. As that figure makes clear, likelihood is still increasing after several hundred iterations.

Perhaps more surprisingly, we often found dramatic changes in accuracy in the order of 5% occuring after several hundred iterations, so we ran 1,000 iterations of EM in all of the experiments described here; each run took approximately 2.5 days computation on a 3.6GHz Pentium 4. It's well-known that accuracy often decreases after the first few EM iterations (which we also observed); however in our experiments we found that performance improves again after 100 iterations and continues improving roughly monotonically. Figure 2 shows how 1-to-1 accuracy varies with iteration during 10 runs from different random starting points. Note that 1-to-1 accuracy at termination ranges from 0.38 to 0.45; a spread of 0.07.

We obtained a dramatic speedup by working directly with probabilities and rescaling after each observation to avoid underflow, rather than working with log probabilities (thanks to Yoshimasa Tsu-

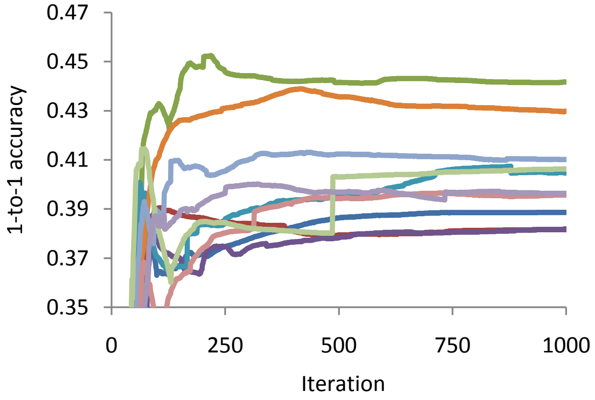


Figure 2: Variation in 1-to-1 accuracy with increasing iterations for 10 EM runs from different random starting points.

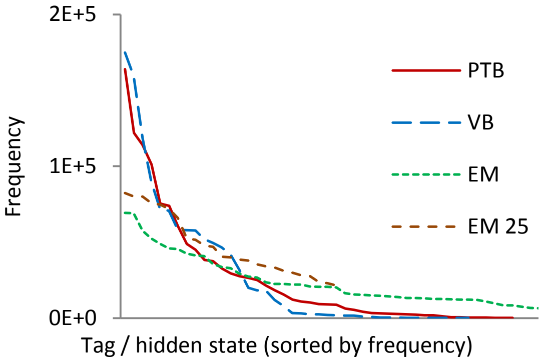


Figure 3: The average number of words labeled with each hidden state or tag for the EM, VB (with $\alpha_x = \alpha_y = 0.1$) and EM-25 estimators (EM-25 is the EM estimator with 25 hidden states).

ruoka for pointing this out). Since we evaluated the accuracy of the estimated tags after each iteration, it was important that decoding be done efficiently as well. While most researchers use Viterbi decoding to find the most likely state sequence, maximum marginal decoding (which labels the observation x_i with the state y_i that maximizes the marginal probability $\mathrm{P}(y_i|\mathbf{x}, \theta, \phi)$) is faster because it re-uses the forward and backward tables already constructed by the Forward-Backward algorithm. Moreover, in separate experiments we found that the maximum marginal state sequence almost always scored higher than the Viterbi state sequence in all of our evaluations, and at modest numbers of iterations (up to 50) often scored more than 5% better.

We also noticed a wide variance in the performance of models due to random initialization (both θ and ϕ are initially jittered to break symmetry); this wide variance was observed with all of the estimators investigated in this paper. This means we cannot compare estimators on the basis of single runs, so we ran each estimator 10 times from different random starting points and report both mean and standard deviation for all scores.

Finally, we also experimented with annealing, in which the parameters θ and ϕ are raised to the power $1/T$, where T is a "temperature" parameter that is slowly lowered toward 1 at each iteration according to some "annealing schedule". We experimented with a variety of starting temperatures and annealing schedules (e.g., linear, exponential, etc), but were unable to find any that produced models whose likelihoods were significantly higher (i.e., the models fit better) than those found without annealing.

The evaluation of the models produced by the EM and other estimators is presented in Table 1. It is difficult to compare these with previous work, but Haghighi and Klein (2006) report that in a completely unsupervised setting, their MRF model, which uses a large set of additional features and a more complex estimation procedure, achieves an average 1-to-1 accuracy of 41.3%. Because they provide no information about the variance in this accuracy it is difficult to tell whether there is a significant difference between their estimator and the EM estimator, but it is clear that when EM is run long enough, the performance of even very simple models like the bitag HMM is better than generally recognized.

As Table 1 makes clear, the EM estimator produces models that are extremely competitive in many-to-1 accuracy and Variation of Information, but are significantly worse in 1-to-1 accuracy. We can understand these results by comparing the distribution of words to hidden states to the distribution of words to POS tags in the gold-standard evaluation corpus. As Figure 3 shows, the distribution of words to POS tags is highly skewed, with just 6 POS tags, NN, IN, NNP, DT, JJ and NNS, accounting for over 55% of the tokens in the corpus. By contrast, the EM distribution is much flatter. This also explains why the many-to-1 accuracy is so much better than the one-to-one accuracy; presumably several hidden

Estimator		1-to-1		Many-to-1		*VI*		$H(T\|Y)$		$H(Y\|T)$	
EM	(50)	0.40	(0.02)	**0.62**	(0.01)	4.46	(0.08)	**1.75**	(0.04)	2.71	(0.06)
VB$(0.1, 0.1)$	(50)	**0.47**	(0.02)	0.50	(0.02)	4.28	(0.09)	2.39	(0.07)	1.89	(0.06)
VB$(0.1, 10^{-4})$	(50)	0.46	(0.03)	0.50	(0.02)	4.28	(0.11)	2.39	(0.08)	1.90	(0.07)
VB$(10^{-4}, 0.1)$	(50)	0.42	(0.02)	0.60	(0.01)	4.63	(0.07)	1.86	(0.03)	2.77	(0.05)
VB$(10^{-4}, 10^{-4})$	(50)	0.42	(0.02)	0.60	(0.01)	4.62	(0.07)	1.85	(0.03)	2.76	(0.06)
GS$(0.1, 0.1)$	(50)	0.37	(0.02)	0.51	(0.01)	5.45	(0.07)	2.35	(0.09)	3.20	(0.03)
GS$(0.1, 10^{-4})$	(50)	0.38	(0.01)	0.51	(0.01)	5.47	(0.04)	2.26	(0.03)	3.22	(0.01)
GS$(10^{-4}, 0.1)$	(50)	0.36	(0.02)	0.49	(0.01)	5.73	(0.05)	2.41	(0.04)	3.31	(0.03)
GS$(10^{-4}, 10^{-4})$	(50)	0.37	(0.02)	0.49	(0.01)	5.74	(0.03)	2.42	(0.02)	3.32	(0.02)
EM	(40)	0.42	(0.03)	0.60	(0.02)	4.37	(0.14)	1.84	(0.07)	2.55	(0.08)
EM	(25)	0.46	(0.03)	0.56	(0.02)	**4.23**	(0.17)	2.05	(0.09)	2.19	(0.08)
EM	(10)	0.41	(0.01)	0.43	(0.01)	4.32	(0.04)	2.74	(0.03)	**1.58**	(0.05)

Table 1: Evaluation of models produced by the various estimators. The values of the Dirichlet prior parameters for α_x and α_y appear in the estimator name for the VB and GS estimators, and the number of hidden states is given in parentheses. Reported values are means over all runs, followed by standard deviations. 10 runs were performed for each of the EM and VB estimators, while 5 runs were performed for the GS estimators. Each EM and VB run consisted of 1,000 iterations, while each GS run consisted of 50,000 iterations. For the estimators with 10 runs, a 3-standard error 95% confidence interval is approximately the same as the standard deviation.

states are being mapped onto a single POS tag. This is also consistent with the fact that the cross-entropy $H(T|Y)$ of tags given hidden states is relatively low (i.e., given a hidden state, the tag is relatively predictable), while the cross-entropy $H(Y|T)$ is relatively high.

4 Bayesian estimation via Gibbs Sampling and Variational Bayes

A Bayesian estimator combines a likelihood term $\mathrm{P}(\mathbf{x}|\theta, \phi)$ and a prior $\mathrm{P}(\theta, \phi)$ to estimate the posterior probability of a model or hidden state sequence. We can use a Bayesian prior to bias our estimator towards models that generate more skewed distributions. Because HMMs (and PCFGs) are products of multinomials, Dirichlet distributions are a particularly natural choice for the priors since they are conjugate to multinomials, which simplifies both the mathematical and computational aspects of the problem. The precise form of the model we investigated is:

$$\begin{array}{lcll}
\theta_y & | & \alpha_y & \sim & \mathrm{Dir}(\alpha_y) \\
\phi_y & | & \alpha_x & \sim & \mathrm{Dir}(\alpha_x) \\
y_i & | & y_{i-1} = y & \sim & \mathrm{Multi}(\theta_y) \\
x_i & | & y_i = y & \sim & \mathrm{Multi}(\phi_y)
\end{array}$$

Informally, α_y controls the sparsity of the state-to-state transition probabilities while α_x controls the sparsity of the state-to-observation emission probabilities. As α_x approaches zero the prior strongly prefers models in which each hidden state emits as few words as possible. This captures the intuition that most word types only belong to one POS, since the minimum number of non-zero state-to-observation transitions occurs when each observation type is emitted from only one state. Similarly, as α_y approaches zero the state-to-state transitions become sparser.

There are two main techniques for Bayesian estimation of such models: Markov Chain Monte Carlo (MCMC) and Variational Bayes (VB). MCMC encompasses a broad range of sampling techniques, including component-wise Gibbs sampling, which is the MCMC technique we used here (Robert and Casella, 2004; Bishop, 2006). In general, MCMC techniques do not produce a single model that characterizes the posterior, but instead produce a stream of samples from the posterior. The application of MCMC techniques, including Gibbs sampling, to HMM inference problems is relatively well-known: see Besag (2004) for a tutorial introduction and Goldwater and Griffiths (2007) for an application of Gibbs sampling to HMM inference for semi-supervised and unsupervised POS tagging.

The Gibbs sampler produces state sequences $\mathbf{y}$ sampled from the posterior distribution:

$$\mathrm{P}(\mathbf{y}|\mathbf{x},\alpha) \propto \int \mathrm{P}(\mathbf{x},\mathbf{y}|\theta,\phi)\mathrm{P}(\theta|\alpha_y)\mathrm{P}(\phi|\alpha_x)\, d\theta\, d\phi$$

Because Dirichlet priors are conjugate to multinomials, it is possible to integrate out the model parameters θ and ϕ to yield the conditional distribution for y_i shown in Figure 4. For each observation x_i in turn, we resample its state y_i conditioned on the states $\mathbf{y}_{-i}$ of the other observations; eventually the distribution of state sequences converges to the desired posterior.

Each iteration of the Gibbs sampler is much faster than the Forward-Backward algorithm (both take time linear in the length of the string, but for an HMM with s hidden states, each iteration of the Gibbs sampler takes $O(s)$ time while each iteration of the Forward-Backward algorithm takes $O(s^2)$ time), so we ran 50,000 iterations of all samplers (which takes roughly the same elapsed time as 1,000 Forward-Backward iterations).

As can be seen from Table 1, the posterior state sequences we obtained are not particularly good. Further, when we examined how the posterior likelihoods varied with increasing iterations of Gibbs sampling, it became apparent that the likelihood was still increasing after 50,000 iterations. Moreover, when comparing posterior likelihoods from different runs with the same prior parameters but different random number seeds, none of the likelihoods crossed, which one would expect if the samplers had converged and were mixing well (Robert and Casella, 2004). Just as with EM, we experimented with a variety of annealing regimes, but were unable to find any which significantly improved accuracy or posterior likelihood.

We also experimented with evaluating state sequences found using maximum posterior decoding (i.e., model parameters are estimated from the posterior sample, and used to perform maximum posterior decoding) rather than the samples from the posterior produced by the Gibbs sampler. We found that the maximum posterior decoding sequences usually scored higher than the posterior samples, but the scores converged after the first thousand iterations. Since the posterior samples are produced as a by-product of Gibbs sampling while maximum posterior decoding requires an additional time consuming step that does not have much impact on scores, we used the posterior samples to produce the results in Table 1.

In contrast to MCMC, Variational Bayesian inference attempts to find the function $Q(\mathbf{y},\theta,\phi)$ that minimizes an upper bound of the negative log likelihood (Jordan et al., 1999):

$$\begin{aligned} & -\log \mathrm{P}(\mathbf{x}) \\ = \ & -\log \int Q(\mathbf{y},\theta,\phi)\frac{\mathrm{P}(\mathbf{x},\mathbf{y},\theta,\phi)}{Q(\mathbf{y},\theta,\phi)}\, d\mathbf{y}\, d\theta\, d\phi \\ \leq & -\int Q(\mathbf{y},\theta,\phi)\log\frac{\mathrm{P}(\mathbf{x},\mathbf{y},\theta,\phi)}{Q(\mathbf{y},\theta,\phi)}\, d\mathbf{y}\, d\theta\, d\phi \quad (3) \end{aligned}$$

The upper bound in (3) is called the *Variational Free Energy*. We make a "mean-field" assumption that the posterior can be well approximated by a factorized model Q in which the state sequence $\mathbf{y}$ does not covary with the model parameters θ, ϕ (this will be true if, for example, there is sufficient data that the posterior distribution has a peaked mode):

$$\mathrm{P}(\mathbf{x},\mathbf{y},\theta,\phi) \ \approx \ Q(\mathbf{y},\theta,\phi) \ = \ Q_1(\mathbf{y})Q_2(\theta,\phi)$$

The calculus of variations is used to minimize the KL divergence between the desired posterior distribution and the factorized approximation. It turns out that if the likelihood and conjugate prior belong to exponential families then the optimal Q_1 and Q_2 do too, and there is an EM-like iterative procedure that finds locally-optimal model parameters (Bishop, 2006).

This procedure is especially attractive for HMM inference, since it involves only a minor modification to the M-step of the Forward-Backward algorithm. MacKay (1997) and Beal (2003) describe Variational Bayesian (VB) inference for HMMs in detail, and Kurihara and Sato (2006) describe VB for PCFGs (which only involves a minor modification to the M-step of the Inside-Outside algorithm). Specifically, the E-step for VB inference for HMMs is the same as in EM, while the M-step is as follows:

$$\begin{aligned} \tilde{\theta}^{(\ell+1)}_{y'|y} &= f(\mathrm{E}[n_{y',y}] + \alpha_y)/f(\mathrm{E}[n_y] + s\alpha_y) \quad (4) \\ \tilde{\phi}^{(\ell+1)}_{x|y} &= f(\mathrm{E}[n_{x,y}] + \alpha_x)/f(\mathrm{E}[n_y] + m\alpha_x) \\ f(v) &= \exp(\psi(v)) \\ \psi(v) &= (v > 7) \ ? \ g(v - \tfrac{1}{2}) : (\psi(v+1) - 1)/v \\ g(x) &\approx \log(x) + 0.04167x^{-2} + 0.00729x^{-4} \\ & \quad +0.00384x^{-6} - 0.00413x^{-8} \ldots \quad (5) \end{aligned}$$

$$\mathrm{P}(y_i|\mathbf{x},\mathbf{y}_{-i},\alpha) \quad \propto \quad \left(\frac{n_{x_i,y_i}+\alpha_x}{n_{y_i}+m\alpha_x}\right)\left(\frac{n_{y_i,y_{i-1}}+\alpha_y}{n_{y_{i-1}}+s\alpha_y}\right)\left(\frac{n_{y_{i+1},y_i}+\mathrm{I}(y_{i-1}=y_i=y_{i+1})+\alpha_y}{n_{y_i}+\mathrm{I}(y_{i-1}=y_i)}\right)$$

Figure 4: The conditional distribution for state y_i used in the Gibbs sampler, which conditions on the states $\mathbf{y}_{-i}$ for all observations *except* x_i. Here m is the number of possible observations (i.e., the size of the vocabulary), s is the number of hidden states and $\mathrm{I}(\cdot)$ is the indicator function (i.e., equal to one if its argument is true and zero otherwise), $n_{x,y}$ is the number of times observation x occurs with state y, $n_{y',y}$ is the number of times state y' follows y, and n_y is the number of times state y occurs; these counts are from $(\mathbf{x}_{-i},\mathbf{y}_{-i})$, i.e., excluding x_i and y_i.

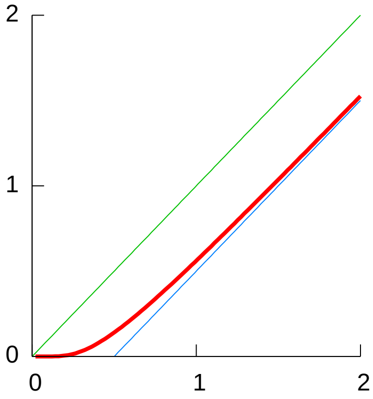


Figure 5: The scaling function $y = f(x) = \exp\psi(x)$ (curved line), which is bounded above by the line $y = x$ and below by the line $y = x - 0.5$.

where ψ is the *digamma function* (the derivative of the log gamma function; (5) gives an asymptotic approximation), and the remaining quantities are just as in the EM updates (2), i.e., $n_{x,y}$ is the number of times observation x occurs with state y, $n_{y',y}$ is the number of times state y' follows y, n_y is the number of occurences of state y, s is the number of hidden states and m is the number of observations; all expectations are taken with respect to the variational parameters $(\tilde{\theta}^{(\ell)}, \tilde{\phi}^{(\ell)})$.

A comparison between (4) and (2) reveals two differences between the EM and VB updates. First, the Dirichlet prior parameters α are added to the expected counts. Second, these posterior counts (which are in fact parameters of the Dirichlet posterior Q_2) are passed through the function $f(v) = \exp\psi(v)$, which is plotted in Figure 5. When $v \gg 0$, $f(v) \approx v - 0.5$, so roughly speaking, VB for multinomials involves adding $\alpha - 0.5$ to the expected counts when they are much larger than zero, where α is the Dirichlet prior parameter. Thus VB can be viewed as a more principled version of the well-known ad hoc technique for approximating Bayesian estimation with EM that involves adding $\alpha - 1$ to the expected counts. However, in the ad hoc approach the expected count plus $\alpha - 1$ may be less than zero, resulting in a value of zero for the corresponding parameter (Johnson et al., 2007; Goldwater and Griffiths, 2007). VB avoids this problem because $f(v)$ is always positive when $v > 0$, even when v is small. Note that because the counts are passed through f, the updated values for $\tilde{\theta}$ and $\tilde{\phi}$ in (4) are in general *not* normalized; this is because the variational free energy is only an upper bound on the negative log likelihood (Beal, 2003).

We found that in general VB performed much better than GS. Computationally it is very similar to EM, and each iteration takes essentially the same time as an EM iteration. Again, we experimented with annealing in the hope of speeding convergence, but could not find an annealing schedule that significantly lowered the variational free energy (the quantity that VB optimizes). While we had hoped that the Bayesian prior would bias VB toward a common solution, we found the same sensitivity to initial conditions as we found with EM, so just as for EM, we ran the estimator for 1,000 iterations with 10 different random initializations for each combination of prior parameters. Table 1 presents the results of VB runs with several different values for the Dirichlet prior parameters. Interestingly, we obtained our best performance on 1-to-1 accuracy when the Dirchlet prior $\alpha_x = 0.1$, a relatively large number, but best performance on many-to-1 accuracy was achieved with a much lower value for the Dirichlet prior, namely $\alpha_x = 10^{-4}$. The Dirichlet prior α_y that controls sparsity of the state-to-state transitions had little effect on the results. We did not have computational resources to fully explore other values for the prior (a set of 10 runs for one set of parameter values takes 25 computer days).

As Figure 3 shows, VB can produce distributions of hidden states that are peaked in the same way that POS tags are. In fact, with the priors used here, VB produces state sequences in which only a subset of the possible HMM states are in fact assigned to observations. This shows that rather than fixing the number of hidden states in advance, the Bayesian prior can determine the number of states; this idea is more fully developed in the *infinite HMM* of Beal et al. (2002) and Teh et al. (2006).

5 Reducing the number of hidden states

EM already performs well in terms of the many-to-1 accuracy, but we wondered if there might be some way to improve its 1-to-1 accuracy and VI score. In section 3 we suggested that one reason for its poor performance in these evaluations is that the distributions of hidden states it finds tend to be fairly flat, compared to the empirical distribution of POS tags. As section 4 showed, a suitable Bayesian prior can bias the estimator towards more peaked distributions, but we wondered if there might be a simpler way of achieving the same result.

We experimented with dramatic reductions in the number of hidden states in the HMMs estimated by EM. This should force the hidden states to be more densely populated and improve 1-to-1 accuracy, even though this means that there will be no hidden states that can possibly map onto the less frequent POS tags (i.e., we will get these words wrong). In effect, we abandon the low-frequency POS tags in the hope of improving the 1-to-1 accuracy of the high-frequency tags.

As Table 1 shows, this markedly improves both the 1-to-1 accuracy and the VI score. A 25-state HMM estimated by EM performs effectively as well as the best VB model in terms of both 1-to-1 accuracy and VI score, and runs 4 times faster because it has only half the number of hidden states.

6 Conclusion and future work

This paper studied why EM seems to do so badly in HMM estimation for unsupervised POS tagging. In fact, we found that it doesn't do so badly at all: the bitag HMM estimated by EM achieves a mean 1-to-1 tagging accuracy of 40%, which is approximately the same as the 41.3% reported by (Haghighi and Klein, 2006) for their sophisticated MRF model.

Then we noted the distribution of words to hidden states found by EM is relatively uniform, compared to the distribution of words to POS tags in the evaluation corpus. This provides an explanation of why the many-to-1 accuracy of EM is so high while the 1-to-1 accuracy and VI of EM is comparatively low. We showed that either by using a suitable Bayesian prior or by simply reducing the number of hidden states it is possible to significantly improve both the 1-to-1 accuracy and the VI score, achieving a 1-to-1 tagging accuracy of 46%.

We also showed that EM and other estimators take much longer to converge than usually thought, and often require several hundred iterations to achieve optimal performance. We also found that there is considerable variance in the performance of all of these estimators, so in general multiple runs from different random starting points are necessary in order to evaluate an estimator's performance.

Finally, there may be more sophisticated ways of improving the 1-to-1 accuracy and VI score than the relatively crude methods used here that primarily reduce the number of available states. For example, we might obtain better performance by using EM to infer an HMM with a large number of states, and then using some kind of distributional clustering to group similar HMM states; these clusters, rather than the underlying states, would be interpreted as the POS tag labels. Also, the Bayesian framework permits a wide variety of different priors besides Dirichlet priors explored here. For example, it should be possible to encode linguistic knowledge such markedness preferences in a prior, and there are other linguistically uninformative priors, such the "entropic priors" of Brand (1999), that may be worth exploring.

Acknowledgements

I would like to thank Microsoft Research for providing an excellent environment in which to conduct this work, and my friends and colleagues at Microsoft Research, especially Bob Moore, Chris Quirk and Kristina Toutanova, for their helpful comments on this paper.

References

Michele Banko and Robert C. Moore. 2004. Part of speech tagging in context. In *Proceedings, 20th International Conference on Computational Linguistics (Coling 2004)*, pages 556–561, Geneva, Switzerland.

M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. 2002. The infinite Hidden Markov Model. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 577–584. The MIT Press.

Matthew J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience unit, University College London.

Julian Besag. 2004. An introduction to Markov Chain Monte Carlo methods. In Mark Johnson, Sanjeev P. Khudanpur, Mari Ostendorf, and Roni Rosenfeld, editors, *Mathematical Foundations of Speech and Language Processing*, pages 247–270. Springer, New York.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

M. Brand. 1999. An entropic estimator for structure discovery. *Advances in Neural Information Processing Systems*, 11:723–729.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the AAAI Workshop on Statistically-Based Natural Language Processing Techniques*, San Jose, CA.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 59–66. Association for Computational Linguistics.

Victoria Fromkin, editor. 2001. *Linguistics: An Introduction to Linguistic Theory*. Blackwell, Oxford, UK.

Sharon Goldwater and Tom Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Joshua Goodman. 2001. A bit of progress in language modeling. *Computer Speech and Language*, 14:403–434.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA, June. Association for Computational Linguistics.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Mark Johnson, Tom Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York. Association for Computational Linguistics.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Sau. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

Kenichi Kurihara and Taisuke Sato. 2006. Variational Bayesian grammar induction for natural language. In *8th International Colloquium on Grammatical Inference*.

David J.C. MacKay. 1997. Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, Cambridge.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Michell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marina Meilă. 2003. Comparing clusterings by the variation of information. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *COLT 2003: The Sixteenth Annual Conference on Learning Theory*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20:155–171.

M. Mitzenmacher. 2003. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods*. Springer.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the*

Association for Computational Linguistics (ACL'05), pages 354–362, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Qin Iris Wang and Dale Schuurmans. 2005. Improved estimation for unsupervised part-of-speech tagging. In *Proceedings of the 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE'2005)*, pages 219–224, Wuhan, China.

Probabilistic Coordination Disambiguation in a Fully-lexicalized Japanese Parser

Daisuke Kawahara
National Institute of Information and
Communications Technology,
3-5 Hikaridai Seika-cho, Soraku-gun,
Kyoto, 619-0289, Japan
dk@nict.go.jp

Sadao Kurohashi
Graduate School of Informatics,
Kyoto University,
Yoshida-Honmachi, Sakyo-ku,
Kyoto, 606-8501, Japan
kuro@i.kyoto-u.ac.jp

Abstract

This paper describes a probabilistic model for coordination disambiguation integrated into syntactic and case structure analysis. Our model probabilistically assesses the parallelism of a candidate coordinate structure using syntactic/semantic similarities and cooccurrence statistics. We integrate these probabilities into the framework of fully-lexicalized parsing based on large-scale case frames. This approach simultaneously addresses two tasks of coordination disambiguation: the detection of coordinate conjunctions and the scope disambiguation of coordinate structures. Experimental results on web sentences indicate the effectiveness of our approach.

1 Introduction

Coordinate structures are a potential source of syntactic ambiguity in natural language. Since their interpretation directly affects the meaning of the text, their disambiguation is important for natural language understanding.

Coordination disambiguation consists of the following two tasks:

- the detection of coordinate conjunctions,
- and finding the scope of coordinate structures.

In English, for example, coordinate structures are triggered by coordinate conjunctions, such as *and* and *or*. In a coordinate structure that consists of more than two conjuncts, commas, which have various usages, also function like coordinate conjunctions. Recognizing true coordinate conjunctions from such possible coordinate conjunctions is a task of coordination disambiguation (Kurohashi, 1995). The other is the task of identifying the range of coordinate phrases or clauses.

Previous work on coordination disambiguation has focused on the task of addressing the scope ambiguity (e.g., (Agarwal and Boggess, 1992; Goldberg, 1999; Resnik, 1999; Chantree et al., 2005)). Kurohashi and Nagao proposed a similarity-based method to resolve both of the two tasks for Japanese (Kurohashi and Nagao, 1994). Their method, however, heuristically detects coordinate conjunctions by considering only similarities between possible conjuncts, and thus cannot disambiguate the following cases[1]:

(1) a. *kanojo-to gakkou-ni itta*
she-cmi school-acc went

(ϕ went to school with her)

b. *kanojo-to watashi-ga goukaku-shita*
she-cnj I-nom passed an exam

(she and I passed an exam)

In sentence (1a), postposition "*to*" is used as a comitative case marker, but in sentence (1b), postposition "*to*" is used as a coordinate conjunction.

To resolve this ambiguity, predicative case frames are required. Case frames describe what kinds of

[1] In this paper, we use the following abbreviations: nom (nominative), acc (accusative), abl (ablative), cmi (comitative), cnj (conjunction) and TM (topic marker).

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 306–314, Prague, June 2007.

Table 1: Case frame examples (Examples are written in English. Numbers following each example represent its frequency.).

	CS	Examples
yaku (1) (broil)	*ga*	I:18, person:15, craftsman:10, · · ·
	wo	bread:2484, meat:1521, cake:1283, · · ·
	de	oven:1630, frying pan:1311, · · ·
yaku (2) (have difficulty)	*ga*	teacher:3, government:3, person:3, · · ·
	wo	fingers:2950
	ni	attack:18, action:15, son:15, · · ·
yaku (3) (burn)	*ga*	maker:1, distributor:1
	wo	data:178, file:107, copy:9, · · ·
	ni	R:1583, CD:664, CDR:3, · · ·
⋮	⋮	⋮
oyogu (1) (swim)	*ga*	dolphin:142, student:50, fish:28, · · ·
	wo	sea:1188, underwater:281, · · ·
	de	crawl:86, breaststroke:49, stroke:24, · · ·
⋮	⋮	⋮
migaku (1) (brush)	*ga*	I:4, man:4, person:4, · · ·
	wo	tooth:5959, molar:27, foretooth:12
	de	brush:38, salt:13, powder:12, · · ·
⋮	⋮	⋮

nouns are related to each predicate. For example, a case frame of "*iku*" (go) has a "*to*" case slot filled with the examples such as "*kanojo*" (she) or human. On the other hand, "*goukaku-suru*" (pass an exam) does not have a "*to*" case slot but does have a "*ga*" case slot filled with "*kanojo*" (she) and "*watashi*" (I). These case frames provide the information for disambiguating the postpositions "*to*" in sentences (1a) and (1b): (1a) is not coordinate and (1b) is coordinate.

This paper proposes a method for integrating coordination disambiguation into probabilistic syntactic and case structure analysis. This method simultaneously addresses the two tasks of coordination disambiguation by utilizing syntactic/semantic parallelism in possible coordinate structures and lexical preferences in large-scale case frames. We use the case frames that were automatically constructed from the web (Table 1). In addition, cooccurrence statistics of coordinate conjuncts are incorporated into this model.

2 Related Work

Previous work on coordination disambiguation has focused mainly on finding the scope of coordinate structures.

Agarwal and Boggess proposed a method for identifying coordinate conjuncts (Agarwal and Boggess, 1992). Their method simply matches parts of speech and hand-crafted semantic tags of the head words of the coordinate conjuncts. They tested their method using the Merck Veterinary Manual and found their method had an accuracy of 81.6%.

Resnik described a similarity-based approach for coordination disambiguation of nominal compounds (Resnik, 1999). He proposed a similarity measure based on the notion of shared information content. He conducted several experiments using the Penn Treebank and reported an F-measure of approximately 70%.

Goldberg applied a cooccurrence-based probabilistic model to determine the attachments of ambiguous coordinate phrases with the form "n1 p n2 cc n3" (Goldberg, 1999). She collected approximately 120K unambiguous pairs of two coordinate words from a raw newspaper corpus for a one-year period and estimated parameters from these statistics. Her method achieved an accuracy of 72% using the Penn Treebank.

Chantree et al. presented a binary classifier for coordination ambiguity (Chantree et al., 2005). Their model is based on word distribution information obtained from the British National Corpus. They achieved an F-measure ($\beta = 0.25$) of 47.4% using their own test set.

The previously described methods focused on coordination disambiguation. Some research has been undertaken that integrated coordination disambiguation into parsing.

Kurohashi and Nagao proposed a Japanese parsing method that included coordinate structure detection (Kurohashi and Nagao, 1994). Their method first detects coordinate structures in a sentence, and then heuristically determines the dependency structure of the sentence under the constraints of the detected coordinate structures. Their method correctly analyzed 97 Japanese sentences out of 150.

Charniak and Johnson used some features of syntactic parallelism in coordinate structures for their MaxEnt reranking parser (Charniak and Johnson, 2005). The reranker achieved an F-measure of 91.0%, which is higher than that of their generative parser (89.7%). However, they used a numerous number of features, and the contribution of the

Table 2: Expressions that indicate coordinate structures.

(a) coordinate noun phrase:
,(comma) *to ya toka katsu oyobi ka aruiwa ...*
(b) coordinate predicative clause:
-shi ga oyobi ka aruiwa matawa ...
(c) incomplete coordinate structure:
,(comma) *oyobi narabini aruiwa ...*

parallelism features is unknown.

Dubey et al. proposed an unlexicalized PCFG parser that modified PCFG probabilities to condition the existence of syntactic parallelism (Dubey et al., 2006). They obtained an F-measure increase of 0.4% over their baseline parser (73.0%). Experiments with a lexicalized parser were not conducted in their work.

A number of machine learning-based approaches to Japanese parsing have been developed. Among them, the best parsers are the SVM-based dependency analyzers (Kudo and Matsumoto, 2002; Sassano, 2004). In particular, Sassano added some features to improve his parser by enabling it to detect coordinate structures (Sassano, 2004). However, the added features did not contribute to improving the parsing accuracy. This failure can be attributed to the inability to consider global parallelism.

3 Coordination Ambiguity in Japanese

In Japanese, the *bunsetsu* is a basic unit of dependency that consists of one or more content words and the following zero or more function words. A bunsetsu corresponds to a base phrase in English and "*eojeol*" in Korean.

Coordinate structures in Japanese are classified into three types. The first type is the *coordinate noun phrase*.

(2) *nagai enpitsu-**to** keshigomu-wo katta*
long pencil-cnj eraser-acc bought

(bought a long pencil and an eraser)

We can find these phrases by referring to the words listed in Table 2-a.

The second type is the *coordinate predicative clause*, in which two or more predicates form a coordinate structure.

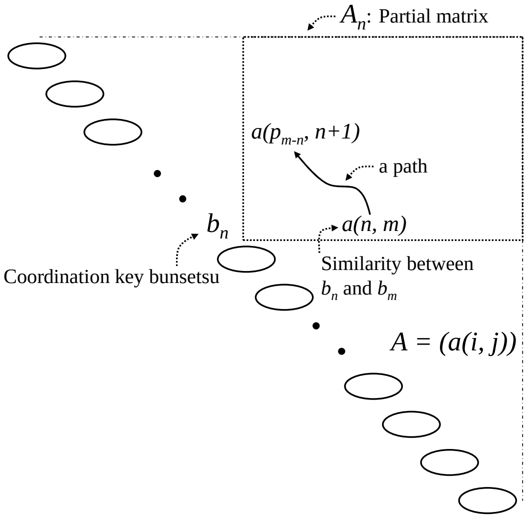


Figure 1: Method using triangular matrix.

(3) *kanojo-to kekkon-**shi** ie-wo katta*
she-cmi married house-acc bought

(married her and bought a house)

We can find these clauses by referring to the words and ending forms listed in Table 2-b.

The third type is the *incomplete coordinate structure*, in which some parts of coordinate predicative clauses are present.

(4) *Tom-wa inu-**wo,** Jim-wa neko-**wo** kau*
Tom-TM dog-acc Jim-TM cat-acc buys

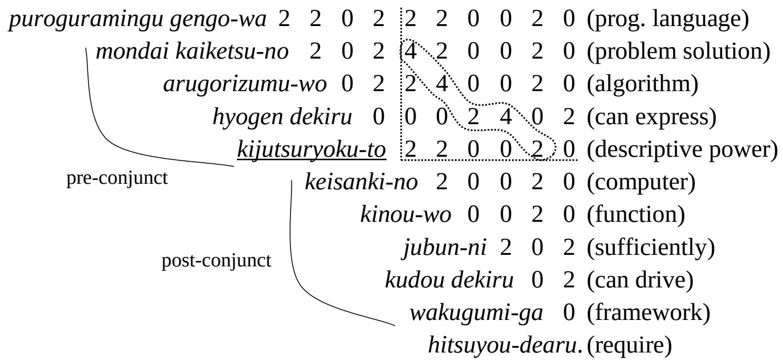
(Tom (buys) a dog, and Jim buys a cat)

We can find these structures by referring to the words listed in Table 2-c and also the correspondence of case-marking postpositions.

For all of these types, we can detect the possibility of a coordinate structure by looking for a *coordination key bunsetsu* that accompanies one of the words listed in Table 2 (in total, we have 52 coordination expressions). That is to say, the left and right sides of a coordination key bunsetsu constitute possible pre- and post-conjuncts, and the key bunsetsu is located at the end of the pre-conjunct. The size of the conjuncts corresponds to the scope of the coordination.

4 Calculating Similarity between Possible Coordinate Conjuncts

We assess the parallelism of potential coordinate structures in a probabilistic parsing model. In this


puroguramingu gengo-wa 2 2 0 2 2 2 0 0 2 0 (prog. language)
mondai kaiketsu-no 2 0 2 4 2 0 0 2 0 (problem solution)
arugorizumu-wo 0 2 2 4 0 0 2 0 (algorithm)
hyogen dekiru 0 0 0 2 4 0 2 (can express)
kijutsuryoku-to 2 2 0 0 2 0 (descriptive power)
pre-conjunct
keisanki-no 2 0 0 2 0 (computer)
kinou-wo 0 0 2 0 (function)
jubun-ni 2 0 2 (sufficiently)
post-conjunct
kudou dekiru 0 2 (can drive)
wakugumi-ga 0 (framework)
hitsuyou-dearu. (require)

(Programming language requires descriptive power to express an algorithm for solving problems and a framework to sufficiently drive functions of a computer.)

Figure 2: Example of calculating path scores.

section, we describe a method for calculating similarities between potential coordinate conjuncts.

To measure the similarity between potential pre- and post-conjuncts, a lot of work on the coordination disambiguation used the similarity between conjoined heads. However, not only the conjoined heads but also other components in conjuncts have some similarity and furthermore structural parallelism. Therefore, we use a method to calculate the similarity between two whole coordinate conjuncts (Kurohashi and Nagao, 1994). The remainder of this section contains a brief description of this method.

To calculate similarity between two series of bunsetsus, a triangular matrix, A, is used (illustrated in Figure 1).

$$A = (a(i,j)) \ \ (0 \leq i \leq l; i \leq j \leq l) \qquad (1)$$

where l is the number of bunsetsus in a sentence, diagonal element $a(i,j)$ is the i-th bunsetsu, and element $a(i,j) \ \ (i < j)$ is the similarity value between bunsetsus b_i and b_j. A similarity value between two bunsetsus is calculated on the basis of POS matching, exact word matching, and their semantic closeness in a thesaurus tree (Kurohashi and Nagao, 1994). We use the *Bunruigoihyo* thesaurus, which contains 96,000 Japanese words (The National Institute for Japanese Language, 2004).

To detect a coordinate structure involving a key bunsetsu, b_n, we consider only a partial matrix (denoted A_n), that is, the upper right part of b_n (Figure 1).

$$A_n = (a(i,j)) \ \ (0 \leq i \leq n; n+1 \leq j \leq l) \qquad (2)$$

To specify correspondences between bunsetsus in potential pre- and post-conjuncts, a path is defined as follows:

$$path ::= (a(p_1, m), \quad a(p_2, m-1), \ \ldots, \\ a(p_{m-n}, n+1)) \qquad (3)$$

where $n+1 \leq m \leq l$, $a(p_1, m) \neq 0$, $p_1 = n$, $p_i \geq p_{i+1}$, $(1 \leq i \leq m-n-1)$.
That is, a path represents a series of elements from a non-zero element in the lowest row in A_n to an element in the leftmost column in A_n. The path has an only element in each column and extends toward the upper left. The series of bunsetsus on the left side of the path and the series under the path are potential conjuncts for key b_n. Figure 2 shows an example of a path.

A path score is defined based on the following criteria:

- the sum of each element's points on the path
- penalty points when the path extends non-diagonally (which causes conjuncts of unbalanced lengths)
- bonus points on expressions signaling the beginning or ending of a coordinate structure, such as "*kaku*" (each) and *nado*" (and so on)
- the total score of the above criteria is divided by the square root of the number of bunsetsus covered by the path for normalization

The score of each path is calculated using a dynamic programming method. We consider each path as a candidate of pre- and post-conjuncts.

5 Integrated Probabilistic Model for Syntactic, Coordinate and Case Structure Analysis

This section describes a method of integrating coordination disambiguation into a probabilistic parsing model. The integrated model is based on a fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis (Kawahara and Kurohashi, 2006b).

5.1 Outline of the Model

This model gives a probability to each possible dependency structure, T, and case structure, L, of the input sentence, S, and outputs the syntactic, coordinate and case structure that have the highest probability. That is to say, the model selects the syntactic structure, T_{best}, and the case structure, L_{best}, that maximize the probability, $P(T, L|S)$:

$$\begin{aligned}(T_{best}, L_{best}) &= \text{argmax}_{(T,L)} P(T, L|S)\\ &= \text{argmax}_{(T,L)} \frac{P(T, L, S)}{P(S)}\\ &= \text{argmax}_{(T,L)} P(T, L, S) \qquad (4)\end{aligned}$$

The last equation is derived because $P(S)$ is constant.

The model considers a clause as a generation unit and generates the input sentence from the end of the sentence in turn. The probability $P(T, L, S)$ is defined as the product of probabilities for generating clause C_i as follows:

$$P(T, L, S) = \prod_{i=1..n} P(C_i, rel_{ih_i}|C_{h_i}) \qquad (5)$$

where n is the number of clauses in S, C_{h_i} is C_i's modifying clause, and rel_{ih_i} is the dependency relation between C_i and C_{h_i}. The main clause, C_n, at the end of a sentence does not have a modifying head, but a virtual clause C_{h_n} = EOS (End Of Sentence) is inserted. Dependency relation rel_{ih_i} is first classified into two types C (coordinate) and D (normal dependency), and C is further divided into five classes according to the binned similarity (path score) of conjuncts. Therefore, rel_{ih_i} can be one of the following six classes.

$$rel_{ih_i} = \{D, C0, C1, C2, C3, C4\} \qquad (6)$$

For instance, $C0$ represents a coordinate relation with a similarity of less than 1, and $C4$ represents a coordinate relation with a similarity of 4 or more.

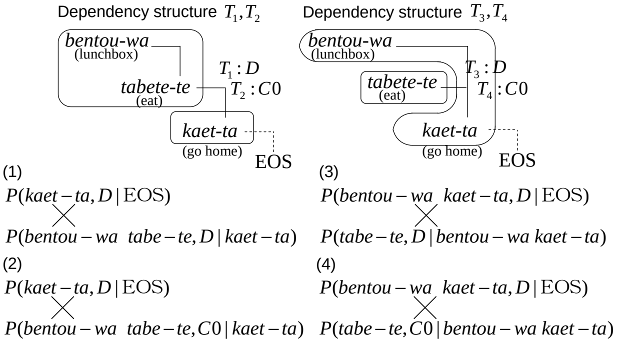

Dependency structure T_1, T_2
bentou-wa
(lunchbox)
tabete-te
(eat)
kaet-ta
(go home)
EOS
$T_1 : D$
$T_2 : C0$
Dependency structure T_3, T_4
bentou-wa
(lunchbox)
tabete-te
(eat)
kaet-ta
(go home)
EOS
$T_3 : D$
$T_4 : C0$
(1)
$P(kaet-ta, D \mid \text{EOS})$
$P(bentou-wa\ \ tabe-te, D \mid kaet-ta)$
(2)
$P(kaet-ta, D \mid \text{EOS})$
$P(bentou-wa\ \ tabe-te, C0 \mid kaet-ta)$
(3)
$P(bentou-wa\ \ kaet-ta, D \mid \text{EOS})$
$P(tabe-te, D \mid bentou-wa\ kaet-ta)$
(4)
$P(bentou-wa\ \ kaet-ta, D \mid \text{EOS})$
$P(tabe-te, C0 \mid bentou-wa\ kaet-ta)$

Figure 3: Example of probability calculation.

For example, consider the sentence shown in Figure 3. There are four possible dependency structures in this figure, and the product of the probabilities for each structure indicated below the tree is calculated. Finally, the model chooses the structure with the highest probability (in this case T_1 is chosen).

Clause C_i is decomposed into its clause type, f_i, (including the predicate's inflection and function words) and its remaining content part $C_i{}'$. Clause C_{h_i} is also decomposed into its content part, $C_{h_i}{}'$, and its clause type, f_{h_i}.

$$\begin{aligned}&P(C_i, rel_{ih_i}|C_{h_i}) = P(C_i{}', f_i, rel_{ih_i}|C_{h_i}{}', f_{h_i})\\ &= P(C_i{}', rel_{ih_i}|f_i, C_{h_i}{}', f_{h_i}) \times P(f_i|C_{h_i}{}', f_{h_i})\\ &\approx P(C_i{}', rel_{ih_i}|f_i, C_{h_i}{}') \times P(f_i|f_{h_i}) \qquad (7)\end{aligned}$$

Equation (7) is derived because the content part, $C_i{}'$, is usually independent of its modifying head type, f_{h_i}, and in most cases, the type, f_i, is independent of the content part of its modifying head, C_{h_i}.

We call $P(C_i{}', rel_{ih_i}|f_i, C_{h_i}{}')$ *generative probability of a case and coordinate structure*, and $P(f_i|f_{h_i})$ *generative probability of a clause type*. The latter is the probability of generating function words including topic markers and punctuation marks, and is estimated using a syntactically annotated corpus in the same way as (Kawahara and Kurohashi, 2006b).

The generative probability of a case and coordinate structure can be rewritten as follows:

$$\begin{aligned}&P(C_i{}', rel_{ih_i}|f_i, C_{h_i}{}')\\ &= P(C_i{}'|rel_{ih_i}, f_i, C_{h_i}{}') \times P(rel_{ih_i}|f_i, C_{h_i}{}')\\ &\approx P(C_i{}'|rel_{ih_i}, f_i, C_{h_i}{}') \times P(rel_{ih_i}|f_i) \qquad (8)\end{aligned}$$

Equation (8) is derived because dependency relations (coordinate or not) heavily depend on modifier's types including coordination keys. We call $P(C_i{}'|rel_{ih_i}, f_i, C_{h_i}{}')$ *generative probability of a case structure*, and $P(rel_{ih_i}|f_i)$ *generative probability of a coordinate structure*. The following two subsections describe these probabilities.

5.2 Generative Probability of Coordinate Structure

The most important feature to decide whether two clauses are coordinate is coordination keys. Therefore, we consider a coordination key, k_i, as clause type f_i. The generative probability of a coordinate structure, $P(rel_{ih_i}|f_i)$, is defined as follows:

$$P(rel_{ih_i}|f_i) = P(rel_{ih_i}|k_i) \tag{9}$$

We classified coordination keys into 52 classes according to the classification proposed by (Kurohashi and Nagao, 1994). If type f_i does not contain a coordination key, the relation is always D (normal dependency), that is $P(rel_{ih_i}|f_i) = P(D|\phi) = 1$.

The generative probability of a coordinate structure was estimated from a syntactically annotated corpus using maximum likelihood. We used the Kyoto Text Corpus (Kurohashi and Nagao, 1998), which consists of 40K Japanese newspaper sentences.

5.3 Generative Probability of Case Structure

We consider that a case structure consists of a predicate, v_i, a case frame, CF_l, and a case assignment, CA_k. Case assignment CA_k represents correspondences between the input case components and the case slots shown in Figure 4. Thus, the generative probability of a case structure is decomposed as follows:

$$\begin{aligned}
&P(C_i{}'|rel_{ih_i}, f_i, C_{h_i}{}') \\
&= P(v_i, CF_l, CA_k|rel_{ih_i}, f_i, C_{h_i}{}') \\
&= P(v_i|rel_{ih_i}, f_i, C_{h_i}{}') \\
&\quad \times P(CF_l|rel_{ih_i}, f_i, C_{h_i}{}', v_i) \\
&\quad \times P(CA_k|rel_{ih_i}, f_i, C_{h_i}{}', v_i, CF_l) \\
&\approx P(v_i|rel_{ih_i}, f_i, w_{h_i}) \\
&\quad \times P(CF_l|v_i) \\
&\quad \times P(CA_k|CF_l, f_i)
\end{aligned} \tag{10}$$

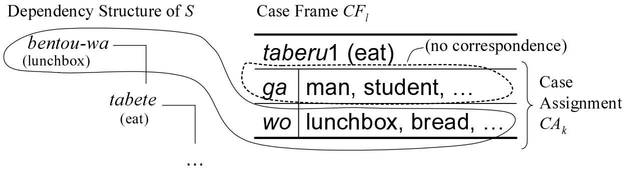


Figure 4: Example of case assignment.

The above approximation is given because it is natural to consider that the predicate v_i depends on its modifying head w_{h_i} instead of the whole modifying clause, that the case frame CF_l only depends on the predicate v_i, and that the case assignment CA_k depends on the case frame CF_l and the clause type f_i.

The generative probabilities of case frames and case assignments are estimated from case frames themselves in the same way as (Kawahara and Kurohashi, 2006b). The remainder of this section describes the generative probability of a predicate, $P(v_i|rel_{ih_i}, f_i, w_{h_i})$.

The generative probability of a predicate captures cooccurrences of coordinate or non-coordinate phrases. This kind of information is not handled in case frames, which aggregate only predicate-argument relations.

The generative probability of a predicate mainly depends on a coordination key in the clause type, f_i, as well as the generative probability of a coordinate structure. We define this probability as follows:

$$P(v_i|rel_{ih_i}, f_i, w_{h_i}) = P(v_i|rel_{ih_i}, k_i, w_{h_i})$$

If $C_i{}'$ is a nominal clause and consists of a noun n_i, we consider the following probability in stead of equation (10):

$$P_n(C_i{}'|rel_{ih_i}, f_i, C_{h_i}{}') \approx P(n_i|rel_{ih_i}, f_i, w_{h_i})$$

This is because a noun does not have a case frame and any case components in the current framework.

To estimate these probabilities, we first applied a conventional parsing system with coordination disambiguation to a huge corpus, and collected coordinate bunsetsus from the parses. We used KNP[2] (Kurohashi and Nagao, 1994) as the parser and a web corpus consisting of 470M Japanese sentences (Kawahara and Kurohashi, 2006a). The generative probability of a predicate was estimated from the

[2] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp-e.html

collected coordinate bunsetsus using maximum likelihood.

5.4 Practical Issue

The proposed model considers all the possible dependency structures including coordination ambiguities. To reduce this high computational cost, we introduced the CKY framework to the search.

Each parameter in the model is smoothed by using several back-off levels in the same way as (Collins, 1999). Smoothing parameters are optimized using a development corpus.

6 Experiments

We evaluated the coordinate structures and dependency structures that were outputted by our model. The case frames used in this paper were automatically constructed from 470M Japanese sentences obtained from the web. Some examples of the case frames are listed in Table 1 (Kawahara and Kurohashi, 2006a).

In this work, the parameters related to unlexical types are calculated from a small tagged corpus of newspaper articles, and lexical parameters are obtained from a huge web corpus. To evaluate the effectiveness of our fully-lexicalized model, our experiments are conducted using web sentences. As the test corpus, we prepared 759 web sentences [3]. The web sentences were manually annotated using the same criteria as the Kyoto Text Corpus. We also used the Kyoto Text Corpus as a development corpus to optimize the smoothing parameters. The system input was automatically tagged using the JUMAN morphological analyzer [4].

We used two baseline systems for comparative purposes: the rule-based dependency parser, KNP (Kurohashi and Nagao, 1994), and the probabilistic model of syntactic and case structure analysis (Kawahara and Kurohashi, 2006b), in which coordination disambiguation is the same as that of KNP.

6.1 Evaluation of Detection of Coordinate Structures

First, we evaluated detecting coordinate structures, namely whether a coordination key bunsetsu triggers a coordinate structure. Table 3 lists the experimental results. The F-measure of our method is slightly higher than that of the baseline method (KNP). In particular, our method achieved good precision.

Table 3: Experimental results of detection of coordinate structures.

	baseline	proposed
precision	366/460 (79.6%)	361/435 (83.0%)
recall	366/447 (81.9%)	361/447 (80.8%)
F-measure	– (80.7%)	– (81.9%)

6.2 Evaluation of Dependency Parsing

Secondly, we evaluated the dependency structures analyzed by the proposed model. Evaluating the scope ambiguity of coordinate structures is subsumed within this dependency evaluation. The dependency structures obtained were evaluated with regard to dependency accuracy — the proportion of correct dependencies out of all dependencies except for the last dependency in the sentence end [5]. Table 4 lists the dependency accuracy. In this table, “syn” represents the rule-based dependency parser, KNP, “syn+case” represents the probabilistic parser of syntactic and case structure (Kawahara and Kurohashi, 2006b), and “syn+case+coord” represents our proposed model. The proposed model significantly outperformed both of the baseline systems (McNemar’s test; $p < 0.01$).

In the table, the dependency accuracies are classified into four types on the basis of the bunsetsu classes (PB: predicate bunsetsu and NB: noun bunsetsu) of a dependent and its head. “syn+case” outperformed “syn”. In particular, the accuracy of predicate-argument relations (“NB→PB”) was improved, but the accuracies of “NB→NB” and “PB→PB” decreased. “syn+case+coord” outperformed the two baselines for all of the types. Not only the accuracy of predicate-argument relations (“NB→PB”) but also the accuracies of coordinate noun/predicate bunsetsus (related to “NB→NB” and “PB→PB”) were improved. These improvements are conduced by the integration of coordination disambiguation and syntactic/case structure analysis.

[3] The test set was not used to construct case frames and estimate probabilities.

[4] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman-e.html

[5] Since Japanese is head-final, the second last bunsetsu unambiguously depends on the last bunsetsu, and the last bunsetsu has no dependency.

Table 4: Experimental results of dependency parsing.

	syn	syn+case	syn+case+coord
all	3,833/4,436 (86.4%)	3,852/4,436 (86.8%)	3,893/4,436 (87.8%)
NB→PB	1,637/1,926 (85.0%)	1,664/1,926 (86.4%)	1,684/1,926 (87.4%)
NB→NB	1,032/1,136 (90.8%)	1,029/1,136 (90.6%)	1,037/1,136 (91.3%)
PB→PB	654/817 (80.0%)	647/817 (79.2%)	659/817 (80.7%)
PB→NB	510/557 (91.6%)	512/557 (91.9%)	513/557 (92.1%)

To compare our results with a state-of-the-art discriminative dependency parser, we input the same test corpus into an SVM-based Japanese dependency parser, CaboCha[6](Kudo and Matsumoto, 2002). Its dependency accuracy was 86.3% (3,829/4,436), which is equivalent to that of "syn" (KNP). This low accuracy is attributed to the out-of-domain training corpus. That is, the parser is trained on a newspaper corpus, whereas the test corpus is obtained from the web, because of the non-availability of a tagged web corpus that is large enough to train a supervised parser.

6.3 Discussion

Figure 5 shows some analysis results, where the dotted lines represent the analysis by the baseline, "syn+case", and the solid lines represent the analysis by the proposed method, "syn+case+coord". These sentences are incorrectly analyzed by the baseline but correctly analyzed by the proposed method. For instance, in sentence (1), the noun phrase coordination of "*apurikeesyon*" (application) and "*doraiba*" (driver) can be correctly analyzed. This is because the case frame of "*insutooru-sareru*" (installed) is likely to generate "*doraiba*", and "*apurikeesyon*" and "*doraiba*" are likely to be coordinated.

One of the causes of errors in dependency parsing is the mismatch between analysis results and annotation criteria. As per the annotation criteria, each bunsetsu has only one modifying head. Therefore, in some cases, even if analysis results are semantically correct, they are judged as incorrect from the viewpoint of the annotation. For example, in sentence (4) in Figure 6, the baseline method, "syn", correctly recognized the head of "*iin-wa*" (commissioner-TM) as "*hirakimasu*" (open). However, the proposed method incorrectly judged it as "*oujite-imasuga*" (offer). Both analysis results can be considered to be semantically correct, but from the viewpoint of our annotation criteria, the latter is not a syntactic relation (i.e., incorrect), but an ellipsis relation. This kind of error is caused by the strong lexical preference considered in our method.

To address this problem, it is necessary to simultaneously evaluate not only syntactic relations but also indirect relations, such as ellipses and anaphora. This kind of mismatch also occurred for the detection of coordinate structures.

Another errors were caused by an inherent characteristic of generative models. Generative models have some advantages, such as their application to language models. However, it is difficult to incorporate various features that seem to be useful for addressing syntactic and coordinate ambiguity. We plan to apply discriminative reranking to the n-best parses produced by our generative model in the same way as (Charniak and Johnson, 2005).

7 Conclusion

This paper has described an integrated probabilistic model for coordination disambiguation and syntactic/case structure analysis. This model takes advantage of lexical preference of a huge raw corpus and large-scale case frames and performs coordination disambiguation and syntactic/case analysis simultaneously. The experiments indicated the effectiveness of our model. Our future work involves incorporating ellipsis resolution to develop an integrated model for syntactic, case, and ellipsis analysis.

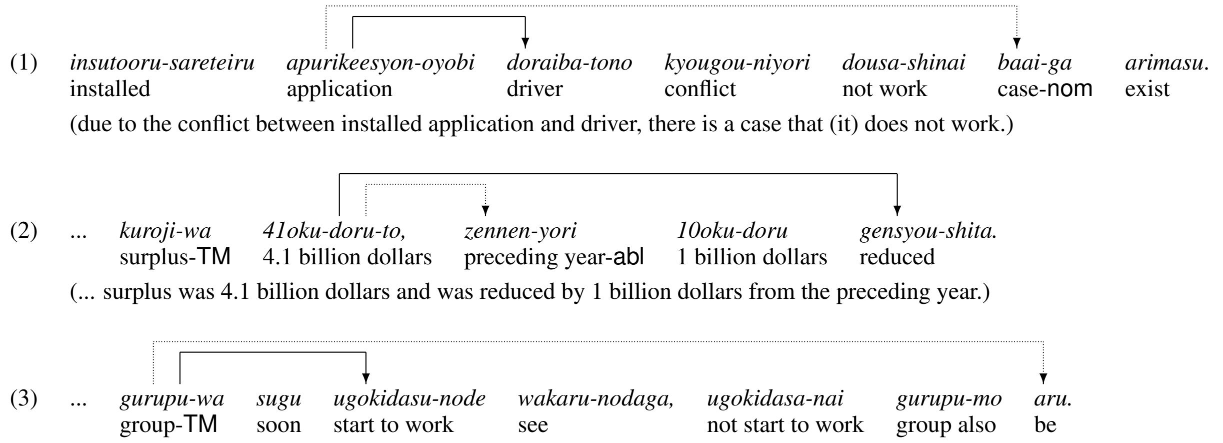
Acknowledgment

This research is partially supported by special coordination funds for promoting science and technology.

References

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of ACL1992*, pages 15–21.

[6] http://chasen.org/˜taku/software/cabocha/

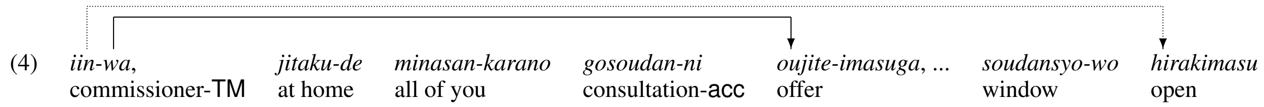


(1) (due to the conflict between installed application and driver, there is a case that (it) does not work.)

(2) (... surplus was 4.1 billion dollars and was reduced by 1 billion dollars from the preceding year.)

(3) (... can see the groups that start to work soon, but there are groups that do not start to work.)

Figure 5: Examples of correct analysis results. The dotted lines represent the analysis by the baseline, "syn+case", and the solid lines represent the analysis by the proposed method, "syn+case+coord".



(the commissioner offers consultation to all of you at home, but opens a window ...)

Figure 6: An example of incorrect analysis results caused by the mismatch between analysis results and annotation criteria.

Francis Chantree, Adam Kilgarriff, Anne de Roeck, and Alistair Wills. 2005. Disambiguating coordinations using word distribution information. In *Proceedings of RANLP2005*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL2005*, pages 173–180.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Amit Dubey, Frank Keller, and Patrick Sturt. 2006. Integrating syntactic priming into an incremental probabilistic parser, with an application to psycholinguistic modeling. In *Proceedings of COLING-ACL2006*, pages 417–424.

Miriam Goldberg. 1999. An unsupervised model for statistically determining coordinate phrase attachment. In *Proceedings of ACL1999*, pages 610–614.

Daisuke Kawahara and Sadao Kurohashi. 2006a. Case frame compilation from the web using high-performance computing. In *Proceedings of LREC2006*.

Daisuke Kawahara and Sadao Kurohashi. 2006b. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of HLT-NAACL2006*, pages 176–183.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of CoNLL2002*, pages 29–35.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proceedings of LREC1998*, pages 719–724.

Sadao Kurohashi. 1995. Analyzing coordinate structures including punctuation in English. In *Proceedings of IWPT1995*, pages 136–147.

Philip Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

Manabu Sassano. 2004. Linear-time dependency analysis for Japanese. In *Proceedings of COLING2004*, pages 8–14.

The National Institute for Japanese Language. 2004. *Bunruigoihyo*. Dainippon Tosho, (In Japanese).

A New Perceptron Algorithm for Sequence Labeling with Non-local Features

Jun'ichi Kazama and Kentaro Torisawa
Japan Advanced Institute of Science and Technology (JAIST)
Asahidai 1-1, Nomi, Ishikawa, 923-1292 Japan
{kazama, torisawa}@jaist.ac.jp

Abstract

We cannot use non-local features with current major methods of sequence labeling such as CRFs due to concerns about complexity. We propose a new perceptron algorithm that can use non-local features. Our algorithm allows the use of all types of non-local features whose values are determined from the sequence and the labels. The weights of local and non-local features are learned together in the training process with guaranteed convergence. We present experimental results from the CoNLL 2003 named entity recognition (NER) task to demonstrate the performance of the proposed algorithm.

1 Introduction

Many NLP tasks such as POS tagging and named entity recognition have recently been solved as sequence labeling. Discriminative methods such as Conditional Random Fields (CRFs) (Lafferty et al., 2001), Semi-Markov Random Fields (Sarawagi and Cohen, 2004), and perceptrons (Collins, 2002a) have been popular approaches for sequence labeling because of their excellent performance, which is mainly due to their ability to incorporate many kinds of overlapping and non-independent features.

However, the common limitation of these methods is that the features are limited to "local" features, which only depend on a very small number of labels (usually two: the previous and the current). Although this limitation makes training and inference tractable, it also excludes the use of possibly useful "non-local" features that are accessible after all labels are determined. For example, non-local features such as "same phrases in a document do not have different entity classes" were shown to be useful in named entity recognition (Sutton and McCallum, 2004; Bunescu and Mooney, 2004; Finkel et al., 2005; Krishnan and Manning, 2006).

We propose a new perceptron algorithm in this paper that can use non-local features along with local features. Although several methods have already been proposed to incorporate non-local features (Sutton and McCallum, 2004; Bunescu and Mooney, 2004; Finkel et al., 2005; Roth and Yih, 2005; Krishnan and Manning, 2006; Nakagawa and Matsumoto, 2006), these present a problem that the types of non-local features are somewhat constrained. For example, Finkel et al. (2005) enabled the use of non-local features by using Gibbs sampling. However, it is unclear how to apply their method of determining the parameters of a non-local model to other types of non-local features, which they did not used. Roth and Yih (2005) enabled the use of hard constraints on labels by using integer linear programming. However, this is equivalent to only allowing non-local features whose weights are fixed to negative infinity. Krishnan and Manning (2006) divided the model into two CRFs, where the second model uses the output of the first as a kind of non-local information. However, it is not possible to use non-local features that depend on the labels of the very candidate to be scored. Nakagawa and Matsumoto (2006) used a Bolzmann distribution to model the correlation of the POS of words having the same lexical form in a document. However, their method can only be applied when there are convenient links such as the same lexical form.

Since non-local features have not yet been extensively investigated, it is possible for us to find new useful non-local features. Therefore, our objective in this study was to establish a framework, where all

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 315–324, Prague, June 2007. ©2007 Association for Computational Linguistics

types of non-local features are allowed.

With non-local features, we cannot use efficient procedures such as forward-backward procedures and the Viterbi algorithm that are required in training CRFs (Lafferty et al., 2001) and perceptrons (Collins, 2002a). Recently, several methods (Collins and Roark, 2004; Daumé III and Marcu, 2005; McDonald and Pereira, 2006) have been proposed with similar motivation to ours. These methods alleviate this problem by using some approximation in perceptron-type learning.

In this paper, we follow this line of research and try to solve the problem by extending Collins' perceptron algorithm (Collins, 2002a). We exploited the not-so-familiar fact that we can design a perceptron algorithm with guaranteed convergence if we can find at least one wrong labeling candidate even if we cannot perform exact inference. We first ran the A* search only using local features to generate n-best candidates (this can be efficiently performed), and then we only calculated the true score with non-local features for these candidates to find a wrong labeling candidate. The second key idea was to update the weights of local features during training if this was necessary to generate sufficiently good candidates. The proposed algorithm combined these ideas to achieve guaranteed convergence and effective learning with non-local features.

The remainder of the paper is organized as follows. Section 2 introduces the Collins' perceptron algorithm. Although this algorithm is the starting point for our algorithm, its baseline performance is not outstanding. Therefore, we present a margin extension to the Collins' perceptron in Section 3. This margin perceptron became the direct basis of our algorithm. We then explain our algorithm for non-local features in Section 4. We report the experimental results using the CoNLL 2003 shared task dataset in Section 6.

2 Perceptron Algorithm for Sequence Labeling

Collins (2002a) proposed an extension of the perceptron algorithm (Rosenblatt, 1958) to sequence labeling. Our aim in sequence labeling is to assign label $y_i \in \mathcal{Y}$ to each word $x_i \in \mathcal{X}$ in a sequence. We denote sequence $x_1, \ldots, x_T$ as $\boldsymbol{x}$ and the corresponding labels as $\boldsymbol{y}$. We assume weight vector $\boldsymbol{\alpha} \in R^d$ and feature mapping Φ that maps each $(\boldsymbol{x}, \boldsymbol{y})$ to feature vector $\Phi(\boldsymbol{x}, \boldsymbol{y}) = (\Phi_1(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_d(\boldsymbol{x}, \boldsymbol{y})) \in R^d$. The model determines the labels by:

$$\boldsymbol{y}' = \mathrm{argmax}_{\boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}|}} \Phi(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha},$$

where $\cdot$ denotes the inner product. The aim of the learning algorithm is to obtain an appropriate weight vector, $\boldsymbol{\alpha}$, given training set $\{(\boldsymbol{x}_1, \boldsymbol{y}_1^*), \cdots, (\boldsymbol{x}_L, \boldsymbol{y}_L^*)\}$.

The learning algorithm, which is illustrated in Collins (2002a), proceeds as follows. The weight vector is initialized to zero. The algorithm passes over the training examples, and each sequence is decoded using the current weights. If $\boldsymbol{y}'$ is not the correct answer $\boldsymbol{y}^*$, the weights are updated according to the following rule.

$$\boldsymbol{\alpha}_{new} = \boldsymbol{\alpha} + \Phi(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi(\boldsymbol{x}, \boldsymbol{y}').$$

This algorithm is proved to converge (i.e., there are no more updates) in the separable case (Collins, 2002a).[1] That is, if there exist weight vector $\boldsymbol{U}$ (with $||\boldsymbol{U}|| = 1$), δ (> 0), and R (> 0) that satisfy:

$$\begin{array}{ll} \forall i, \forall \boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}_i|} & \Phi(\boldsymbol{x_i}, \boldsymbol{y_i}^*) \cdot \boldsymbol{U} - \Phi(\boldsymbol{x_i}, \boldsymbol{y}) \cdot \boldsymbol{U} \geq \delta, \\ \forall i, \forall \boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}_i|} & ||\Phi(\boldsymbol{x_i}, \boldsymbol{y_i}^*) - \Phi(\boldsymbol{x_i}, \boldsymbol{y})|| \leq R, \end{array}$$

the number of updates is at most R^2/δ^2.

The perceptron algorithm only requires one candidate $\boldsymbol{y}'$ for each sequence $\boldsymbol{x_i}$, unlike the training of CRFs where all possible candidates need to be considered. This inherent property is the key to training with non-local features. However, note that the tractability of learning and inference relies on how efficiently $\boldsymbol{y}'$ can be found. In practice, we can find $\boldsymbol{y}'$ efficiently using a Viterbi-type algorithm only when the features are all local, i.e., $\Phi_s(\boldsymbol{x}, \boldsymbol{y})$ can be written as the sum of (two label) local features ϕ_s as $\Phi_s(\boldsymbol{x}, \boldsymbol{y}) = \sum_i^T \phi_s(\boldsymbol{x}, y_{i-1}, y_i)$. This locality constraint is also required to make the training of CRFs tractable (Lafferty et al., 2001).

One problem with the perceptron algorithm described so far is that it offers no treatment for overfitting. Thus, Collins (2002a) also proposed an averaged perceptron, where the final weight vector is

[1] Collins (2002a) also provided proof that guaranteed "good" learning for the non-separable case. However, we have only considered the separable case throughout the paper.

Algorithm 3.1: Perceptron with margin for sequence labeling (parameters: C)

```
$\boldsymbol{\alpha} \leftarrow \mathbf{0}$
until no more updates do
  for $i \leftarrow 1$ to $L$ do
    $\boldsymbol{y}' = \text{argmax}_{\boldsymbol{y}} \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y}} \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    if $\boldsymbol{y}' \neq \boldsymbol{y}_i^*$ then
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi(\boldsymbol{x}_i, \boldsymbol{y}')$
    else if $\Phi(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x}_i, \boldsymbol{y}'') \cdot \boldsymbol{\alpha} \leq C$ then
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi(\boldsymbol{x}_i, \boldsymbol{y}'')$
```

the average of all weight vectors during training. Howerver, we found in our experiments that the averaged perceptron performed poorly in our setting. We therefore tried to make the perceptron algorithm more robust to overfitting. We will describe our extension to the perceptron algorithm in the next section.

3 Margin Perceptron Algorithm for Sequence Labeling

We extended a perceptron with a margin (Krauth and Mézard, 1987) to sequence labeling in this study, as Collins (2002a) extended the perceptron algorithm to sequence labeling.

In the case of sequence labeling, the margin is defined as:

$$\gamma(\boldsymbol{\alpha}) = \min_{\boldsymbol{x}_i} \min_{\boldsymbol{y} \neq \boldsymbol{y}_i^*} \frac{\Phi(\boldsymbol{x_i}, \boldsymbol{y_i}^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x_i}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}}{||\boldsymbol{\alpha}||}$$

Assuming that the best candidate, $\boldsymbol{y}'$, equals the correct answer, $\boldsymbol{y}^*$, the margin can be re-written as:

$$= \min_{\boldsymbol{x}_i} \frac{\Phi(\boldsymbol{x_i}, \boldsymbol{y_i}^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x_i}, \boldsymbol{y''}) \cdot \boldsymbol{\alpha}}{||\boldsymbol{\alpha}||},$$

where $\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y}} \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$. Using this relation, the resulting algorithm becomes Algorithm 3.1. The algorithm tries to enlarge the margin as much as possible, as well as make the best scoring candidate equal the correct answer.

Constant C in Algorithm 3.1 is a tunable parameter, which controls the trade-off between the margin and convergence time. Based on the proofs in Collins (2002a) and Li et al. (2002), we can prove that the algorithm converges within $(2C + R^2)/\delta^2$ updates and that $\gamma(\boldsymbol{\alpha}) \geq \delta C/(2C + R^2) = (\delta/2)(1 - (R^2/(2C + R^2)))$ after training. As can be seen, the margin approaches at least half of true margin δ (at the cost of infinite training time), as $C \rightarrow \infty$.

Note that if the features are all local, the second-best candidate (generally n-best candidates) can also be found efficiently by using an A* search that uses the best scores calculated during a Viterbi search as the heuristic estimation (Soong and Huang, 1991).

There are other methods for improving robustness by making margin larger for the structural output problem. Such methods include ALMA (Gentile, 2001) used in (Daumé III and Marcu, 2005)[2], MIRA (Crammer et al., 2006) used in (McDonald et al., 2005), and Max-Margin Markov Networks (Taskar et al., 2003). However, to the best of our knowledge, there has been no prior work that has applied a perceptron with a margin (Krauth and Mézard, 1987) to structured output.[3] Our method described in this section is one of the easiest to implement, while guaranteeing a large margin. We found in the experiments that our method outperformed the Collins' averaged perceptron by a large margin.

4 Algorithm

4.1 Definition and Basic Idea

Having described the basic perceptron algorithms, we will know explain our algorithm that learns the weights of local and non-local features in a unified way.

Assume that we have local features and non-local features. We use the superscript, l, for local features as $\Phi_i^l(\boldsymbol{x}, \boldsymbol{y})$ and g for non-local features as $\Phi_i^g(\boldsymbol{x}, \boldsymbol{y})$. Then, feature mapping is written as $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) = \Phi^l(\boldsymbol{x}, \boldsymbol{y}) + \Phi^g(\boldsymbol{x}, \boldsymbol{y}) = (\Phi_1^l(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_n^l(\boldsymbol{x}, \boldsymbol{y}), \Phi_{n+1}^g(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_d^g(\boldsymbol{x}, \boldsymbol{y}))$. Here, we define:

$$\begin{aligned} \Phi^l(\boldsymbol{x}, \boldsymbol{y}) &= (\Phi_1^l(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_n^l(\boldsymbol{x}, \boldsymbol{y}), 0, \cdots, 0) \\ \Phi^g(\boldsymbol{x}, \boldsymbol{y}) &= (0, \cdots, 0, \Phi_{n+1}^g(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_d^g(\boldsymbol{x}, \boldsymbol{y})) \end{aligned}$$

Ideally, we want to determine the labels using the whole feature set as:

$$\boldsymbol{y}' = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}|}} \Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}.$$

[2] (Daumé III and Marcu, 2005) also presents the method using the averaged perceptron (Collins, 2002a)

[3] For re-ranking problems, Shen and Joshi (2004) proposed a perceptron algorithm that also uses margins. The difference is that our algorithm trains the sequence labeler itself and is much simpler because it only aims at labeling.

Algorithm 4.1: Candidate algorithm (parameters: n, C)

```
$\boldsymbol{\alpha} \leftarrow \mathbf{0}$
until no more updates do
  for $i \leftarrow 1$ to $L$ do
    $\{\boldsymbol{y}^n\} = \text{n-best}_{\boldsymbol{y}} \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}' = \text{argmax}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    if $\boldsymbol{y}' \neq \boldsymbol{y_i}^*$
        & $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}') \cdot \boldsymbol{\alpha} \leq C$ then
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}')$
    else if $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}'') \cdot \boldsymbol{\alpha} \leq C$ then
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}'')$
```

Algorithm 4.2: Perceptron with local and non-local features (parameters: n, C^a, C^l)

```
$\boldsymbol{\alpha} \leftarrow \mathbf{0}$
until no more updates do
  for $i \leftarrow 1$ to $L$ do
    $\{\boldsymbol{y}^n\} = \text{n-best}_{\boldsymbol{y}} \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}' = \text{argmax}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    if $\boldsymbol{y}' \neq \boldsymbol{y}_i^*$
        & $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}') \cdot \boldsymbol{\alpha} \leq C^a$ then
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}')$                    (A)
    else if $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}'') \cdot \boldsymbol{\alpha} \leq C^a$ then
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}'')$                    (A)
    else
      (B)  if $\boldsymbol{y}^1 \neq \boldsymbol{y_i}^*$ then ($\boldsymbol{y}^1$ represents the best in $\{\boldsymbol{y}^n\}$)
             $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}^1)$
           else if $\Phi^l(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}^2) \cdot \boldsymbol{\alpha} \leq C^l$ then
             $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}^2)$
```

However, if there are non-local features, it is impossible to find the highest scoring candidate efficiently, since we cannot use the Viterbi algorithm. Thus, we cannot use the perceptron algorithms described in the previous sections. The training of CRFs is also intractable for the same reason.

To deal with this problem, we first relaxed our objective. The modified objective was to find a good model from those with the form:

$$\begin{aligned} \{\boldsymbol{y}^n\} &= \text{n-best}_{\boldsymbol{y}} \Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha} \\ \boldsymbol{y}' &= \text{argmax}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}, \quad (1) \end{aligned}$$

That is, we first generate n-best candidates $\{\boldsymbol{y}^n\}$ under the local model, $\Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$. This can be done efficiently using the A* algorithm. We then find the best scoring candidate under the total model, $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$, only from these n-best candidates. If n is moderately small, this can also be done in a practical amount of time.

This resembles the re-ranking approach (Collins and Duffy, 2002; Collins, 2002b). However, unlike the re-ranking approach, the local model, $\Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$, and the total model, $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$, correlate since they share a part of the vector and are trained at the same time in our algorithm. The re-ranking approach has the disadvantage that it is necessary to use different training corpora for the first model and for the second, or to use cross validation type training, to make the training for the second meaningful. This reduces the effective size of training data or increases training time substantially. On the other hand, our algorithm has no such disadvantage.

However, we are no longer able to find the highest scoring candidate under $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$ exactly with this approach. We cannot thus use the perceptron algorithms directly. However, by examining the proofs in Collins (2002a), we can see that the essential condition for convergence is that the weights are always updated using some $\boldsymbol{y}$ ($\neq \boldsymbol{y}^*$) that satisfies:

$$\Phi(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha} \leq 0$$

($\leq C$ in the case of a perceptron with a margin). (2)

That is, $\boldsymbol{y}$ does not necessarily need to be the exact best candidate or the exact second-best candidate. The algorithm also converges in a finite number of iterations even with Eq. (1) as long as Eq. (2) is satisfied.

4.2 Candidate Algorithm

The algorithm we came up with first based on the above idea, is Algorithm 4.1. We first find the n-best candidates using the local model, $\Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$. At this point, we can determine the value of the non-local features, $\Phi^g(\boldsymbol{x}, \boldsymbol{y})$, to form the whole feature vector, $\Phi^a(\boldsymbol{x}, \boldsymbol{y})$, for the n-best candidates. Next, we re-score and sort them using the total model, $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$, to find a candidate that violates the margin condition. We call this algorithm the "candidate algorithm". After the training has finished, $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha} > C$ is guaranteed for all $(\boldsymbol{x}_i, \boldsymbol{y})$ where $\boldsymbol{y} \in \{\boldsymbol{y}^n\}, \boldsymbol{y} \neq \boldsymbol{y}^*$. At first glance, this seems sufficient condition for good models. However, this is not true because if $\boldsymbol{y}^* \notin \{\boldsymbol{y}^n\}$, the inference defined by Eq. (1) is not guaranteed to find the correct answer, $\boldsymbol{y}^*$. In fact, this algorithm does not work well with non-local features as we found in the experiments.

4.3 Final Algorithm

Our idea for improving the above algorithm is that the local model, $\Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$, must at least be so good that $\boldsymbol{y}^* \in \{\boldsymbol{y}^n\}$. To achieve this, we added a modification term that was intended to improve the local model when the local model was not good enough even when the total model was good enough.

The final algorithm resulted in Algorithm 4.2. As can be seen, the part marked (B) has been added. We call this algorithm the "proposed algorithm". Note that the algorithm prioritizes the update of the total model, (A), over that of the local model, (B), although the opposite is also possible. Also note that the update of the local model in (B) is "aggressive" since it updates the weights until the best candidate output by the local model becomes the correct answer and satisfies the margin condition. A "conservative" updating, where we cease the update when the n-best candidates contain the correct answer, is also possible from our idea above. We made these choices since they worked better than the other alternatives.

The tunable parameters are the local margin parameter, C^l, the total margin parameter, C^a, and n for the n-best search. We used $C = C^l = C^a$ in this study to reduce the search space.

We can prove that the algorithm in Algorithm 4.2 also converges in a finite number of iterations. It converges within $(2C + R^2)/\delta^2$ updates, assuming that there exist weight vector $\boldsymbol{U}^l$ (with $||\boldsymbol{U}^l|| = 1$ and $U_i^l = 0 \ \ (n+1 \leq i \leq d)$), δ (> 0), and R (> 0) that satisfy:

$$\forall i, \forall \boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}_i|} \ \ \Phi^l(\boldsymbol{x_i}, \boldsymbol{y_i}^*) \cdot \boldsymbol{U}^l - \Phi^l(\boldsymbol{x_i}, \boldsymbol{y}) \cdot \boldsymbol{U}^l \geq \delta,$$

$$\forall i, \forall \boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}_i|} \ \ ||\Phi^a(\boldsymbol{x_i}, \boldsymbol{y_i}^*) - \Phi^a(\boldsymbol{x_i}, \boldsymbol{y})|| \leq R.$$

In addition, we can prove that $\gamma'(\boldsymbol{\alpha}) \geq \delta C/(2C + R^2)$ for the margin after convergence, where $\gamma'(\boldsymbol{\alpha})$ is defined as:

$$\min_{\boldsymbol{x}_i} \min_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}, \neq \boldsymbol{y}_i^*} \frac{\Phi^a(\boldsymbol{x_i}, \boldsymbol{y_i}^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x_i}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}}{||\boldsymbol{\alpha}||}$$

See Appendix A for the proofs.

We also incorporated the idea behind Bayes point machines (BPMs) (Herbrich and Graepel, 2000) to improve the robustness of our method further. BPMs try to cancel out overfitting caused by the order of examples, by training several models by shuffling the training examples.[4] However, it is very time consuming to run the complete training process several times. We thus ran the training in only one pass over the shuffled examples several times, and used the averaged output weight vectors as a new initial weight vector, because we thought that the early part of training would be more seriously affected by the order of examples. We call this "BPM initialization". [5]

5 Named Entity Recognition and Non-Local Features

We evaluated the performance of the proposed algorithm using the named entity recognition task. We adopted IOB (IOB2) labeling (Ramshaw and Marcus, 1995), where the first word of an entity of class "C" is labeled "B-C", the words in the entity are labeled "I-C", and other words are labeled "O".

We used non-local features based on Finkel et al. (2005). These features are based on observations such as "same phrases in a document tend to have the same entity class" (phrase consistency) and "a sub-phrase of a phrase tends to have the same entity class as the phrase" (sub-phrase consistency). We also implemented the "majority" version of these features as used in Krishnan and Manning (2006). In addition, we used non-local features, which are based on the observation that "entities tend to have the same entity class if they are in the same conjunctive or disjunctive expression" as in "$\cdots$ in U.S., EU, and Japan" (conjunction consistency). This type of non-local feature was not used by Finkel et al. (2005) or Krishnan and Manning (2006).

6 Experiments

6.1 Data and Setting

We used the English dataset of the CoNLL 2003 named entity shared task (Tjong et al., 2003) for the experiments. It is a corpus of English newspaper articles, where four entity classes, PER, LOC, ORG, and MISC are annotated. It consists of training, development, and testing sets (14,987, 3,466,

[4] The results for the perceptron algorithms generally depend on the order of the training examples.

[5] Note that we can prove that the perceptron algorithms converge even though the weight vector is not initialized as $\boldsymbol{\alpha} = \boldsymbol{0}$.

and 3,684 sentences, respectively). Automatically assigned POS tags and chunk tags are also provided. The CoNLL 2003 dataset contains document boundary markers. We concatenated the sentences in the same document according to these markers.[6] This generated 964 documents for the training set, 216 documents for the development set, and 231 documents for the testing set. The documents generated as above become the sequence, $\boldsymbol{x}$, in the learning algorithms.

We first evaluated the baseline performance of a CRF model, the Collins' perceptron, and the Collins' averaged perceptron, as well as the margin perceptron, with only local features. We next evaluated the performance of our perceptron algorithm proposed for non-local features.

We used the local features summarized in Table 1, which are similar to those used in other studies on named entity recognition. We omitted features whose surface part listed in Table 1 occurred less than twice in the training corpus.

We used CRF++ (ver. 0.44)[7] as the basis of our implementation. We implemented scaling, which is similar to that for HMMs (see such as (Rabiner, 1989)), in the forward-backward phase of CRF training to deal with very long sequences due to sentence concatenation.[8]

We used Gaussian regularization (Chen and Rosenfeld, 2000) for CRF training to avoid overfitting. The parameter of the Gaussian, σ^2, was tuned using the development set. We also tuned the margin parameter, C, for the margin perceptron algorithm.[9] The convergence of CRF training was determined by checking the log-likelihood of the model. The convergence of perceptron algorithms was determined by checking the per-word labeling error, since the number of updates was not zero even after a large number of iterations in practice. We stopped training when the relative change in these values became less than a pre-defined threshold (0.0001) for at least three iterations.

Table 1: Local features used. The value of a node feature is determined from the current label, y_0, and a surface feature determined only from $\boldsymbol{x}$. The value of an edge feature is determined by the previous label, y_{-1}, the current label, y_0, and a surface feature. Used surface features are the word (w), the downcased word (wl), the POS tag (pos), the chunk tag (chk), the prefix of the word of length n (pn), the suffix (sn), the word form features: 2d - cp (these are based on (Bikel et al., 1999)), and the gazetteer features: go for ORG, gp for PER, and gm for MISC. These represent the (longest) match with an entry in the gazetteer by using IOB2 tags.

Node features: $\{\text{""}, \mathrm{x}_{-2}, \mathrm{x}_{-1}, \mathrm{x}_{0}, \mathrm{x}_{+1}, \mathrm{x}_{+2}\} \times y_0$
x =, w, wl, pos, chk, p1, p2, p3, p4, s1, s2, s3, s4, 2d, 4d, d&a, d&-, d&/, d&,, d&., n, ic, ac, l, cp, go, gp, gm
Edge features: $\{\text{""}, \mathrm{x}_{-2}, \mathrm{x}_{-1}, \mathrm{x}_{0}, \mathrm{x}_{+1}, \mathrm{x}_{+2}\} \times y_{-1} \times y_0$
x =, w, wl, pos, chk, p1, p2, p3, p4, s1, s2, s3, s4, 2d, 4d, d&a, d&-, d&/, d&,, d&., n, ic, ac, l, cp, go, gp, gm
Bigram node features: $\{\mathrm{x}_{-2}\mathrm{x}_{-1}, \mathrm{x}_{-1}\mathrm{x}_{0}, \mathrm{x}_{0}\mathrm{x}_{+1}\} \times y_0$
x = wl, pos, chk, go, gp, gm
Bigram edge features: $\{\mathrm{x}_{-2}\mathrm{x}_{-1}, \mathrm{x}_{-1}\mathrm{x}_{0}, \mathrm{x}_{0}\mathrm{x}_{+1}\} \times y_{-1} \times y_0$
x = wl, pos, chk, go, gp, gm

We used $n = 20$ (n of the n-best) for training since we could not use too a large n because it would have slowed down training. However, we could examine a larger n during testing, since the testing time did not dominate the time for the experiment. We found an interesting property for n in our preliminary experiment. We found that an even larger n in testing (written as n') achieved higher accuracy, although it is natural to assume that the same n that was used in training would also be appropriate for testing. We thus used $n' = 100$ to evaluate performance during parameter tuning. After finding the best C with $n' = 100$, we varied n' to investigate its

[6] We used sentence concatenation even when only using local features, since we found it does not degrade accuracy (rather we observed a slight increase).

[7] http://chasen.org/˜taku/software/CRF++

[8] We also replaced the optimization module in the original package with that used in the Amis maximum entropy estimator (http://www-tsujii.is.s.u-tokyo.ac.jp/amis) since we encountered problems with the provided module in some cases.

[9] For the Gaussian parameter, we tested {13, 25, 50, 100, 200, 400, 800} (the accuracy did not change drastically among these values and it seems that there is no accuracy hump even if we use smaller values). We tested {500, 1000, 1414, 2000, 2828, 4000, 5657, 8000, 11313, 16000, 32000} for the margin parameters.

Table 2: Summary of performance (F_1).

Method	dev	test	C (or σ^2)
local features			
CRF	**91.10**	**86.26**	100
Perceptron	89.01	84.03	-
Averaged perceptron	89.32	84.08	-
Margin perceptron	**90.98**	**85.64**	11313
+ non-local features			
Candidate ($n' = 100$)	90.71	84.90	4000
Proposed ($n' = 100$)	**91.95**	**86.30**	5657

Table 3: Effect of n'.

Method	dev	test	C
Proposed ($n' = 20$)	91.76	86.19	5657
Proposed ($n' = 100$)	91.95	86.30	5657
Proposed ($n' = 400$)	92.13	86.39	5657
Proposed ($n' = 800$)	92.09	86.39	5657
Proposed ($n' = 1600$)	92.13	**86.46**	5657
Proposed ($n' = 6400$)	**92.19**	86.38	5657

effects further.

6.2 Results

Table 2 compares the results. CRF outperformed the perceptron by a large margin. Although the averaged perceptron outperformed the perceptron, the improvement was slight. However, the margin perceptron greatly outperformed compared to the averaged perceptron. Yet, CRF still had the best baseline performance with only local features.

The proposed algorithm with non-local features improved the performance on the test set by 0.66 points over that of the margin perceptron without non-local features. The row "Candidate" refers to the candidate algorithm (Algorithm 4.1). From the results for the candidate algorithm, we can see that the modification part, (B), in Algorithm 4.2 was essential to make learning with non-local features effective.

We next examined the effect of n'. As can be seen from Table 3, an n' larger than that for training yields higher performance. The highest performance with the proposed algorithm was achieved when $n' = 6400$, where the improvement due to non-local features became 0.74 points.

The performance of the related work (Finkel et al., 2005; Krishnan and Manning, 2006) is listed in Table 4. We can see that the final performance of our algorithm was worse than that of the related work.

We changed the experimental setting slightly to investigate our algorithm further. Instead of the POS/chunk tags provided in the CoNLL 2003 dataset, we used the tags assigned by TagChunk (Daumé III and Marcu, 2005)[10] with the intention of using more accurate tags. The results with this setting are summarized in Table 5. Performance was better than that in the previous experiment for all algorithms. We think this was due to the quality of the POS/chunk tags. It is interesting that the effect of non-local features rose to 0.93 points with $n' = 6400$, even though the baseline performance was also improved. The resulting performance of the proposed algorithm with non-local features is higher than that of Finkel et al. (2005) and comparable with that of Krishnan and Manning (2006). This comparison, of course, is not fair because the setting was different. However, we think the results demonstrate a potential of our new algorithm.

Table 4: The performance of the related work.

Method	dev	test
Finkel et al., 2005 (Finkel et al., 2005)		
baseline CRF	-	85.51
+ non-local features	-	86.86
Krishnan and Manning, 2006 (Krishnan and Manning, 2006)		
baseline CRF	-	85.29
+ non-local features	-	87.24

Table 5: Summary of performance with POS/chunk tags by TagChunk.

Method	dev	test	C (or σ^2)
local features			
CRF	**91.39**	**86.30**	200
Perceptron	89.36	84.35	-
Averaged perceptron	89.76	84.50	-
Margin perceptron	**91.06**	**86.24**	32000
+ non-local features			
Proposed ($n' = 100$)	92.23	87.04	5657
Proposed ($n' = 6400$)	**92.54**	**87.17**	5657

The effect of BPM initialization was also examined. The number of BPM runs was 10 in this experiment. The performance of the proposed algorithm dropped from $91.95/86.30$ to $91.89/86.03$ without BPM initialization as expected in the setting of the experiment of Table 2. The performance of the margin perceptron, on the other hand, changed from $90.98/85.64$ to $90.98/85.90$ without BPM initialization. This result was unexpected from the result of our preliminary experiment. However, the performance was changed from $91.06/86.24$ to

[10] http://www.cs.utah.edu/~hal/TagChunk/

Table 6: Comparison with re-ranking approach.

Method	dev	test	C
local features			
Margin Perceptron	91.06	86.24	32000
+ non-local features			
Re-ranking 1 ($n' = 100$)	91.62	86.57	4000
Re-ranking 1 ($n' = 80$)	**91.71**	**86.58**	4000
Re-ranking 2 ($n' = 100$)	92.08	86.86	16000
Re-ranking 2 ($n' = 800$)	**92.26**	**86.95**	16000
Proposed ($n' = 100$)	92.23	87.04	5657
Proposed ($n' = 6400$)	**92.54**	**87.17**	5657

Table 7: Comparison of training time ($C = 5657$).

Method	dev	test	time (sec.)
local features			
Margin Perceptron	91.04	86.28	15,977
+ non-local features			
Re-ranking 1 ($n' = 100$)	91.48	86.53	86,742
Re-ranking 2 ($n' = 100$)	92.02	86.85	112,138
Proposed ($n' = 100$)	92.23	87.04	28,880

$91.17/86.08$ (i.e., dropped for the evaluation set as expected), in the setting of the experiment of Table 5. Since the effect of BPM initialization is not conclusive only from these results, we need more experiments on this.

6.3 Comparison with re-ranking approach

Finally, we compared our algorithm with the reranking approach (Collins and Duffy, 2002; Collins, 2002b), where we first generate the n-best candidates using a model with only local features (the first model) and then re-rank the candidates using a model with non-local features (the second model).

We implemented two re-ranking models, "reranking 1" and "re-ranking 2". These models differ in how to incorporate the local information in the second model. "re-ranking 1" uses the score of the first model as a feature in addition to the non-local features as in Collins (2002b). "re-ranking 2" uses the same local features as the first model[11] in addition to the non-local features. The first models were trained using the margin perceptron algorithm in Algorithm 3.1. The second models were trained using the algorithm, which is obtained by replacing $\{\boldsymbol{y}^n\}$ with the n-best candidates by the first model. The first model used to generate n-best candidates for the development set and the test set was trained using the whole training data. However, CRFs or perceptrons generally have nearly zero error on the training data, although the first model should mis-label to some extent to make the training of the second model meaningful. To avoid this problem, we adopt cross-validation training as used in Collins (2002b). We split the training data into 5 sets. We then trained five first models using 4/5 of the data, each of which was used to generate n-best candidates for the remaining 1/5 of the data.

As in the previous experiments, we tuned C using the development set with $n' = 100$ and then tested other values for n'. Table 6 shows the results. As can be seen, re-ranking models were outperformed by our proposed algorithm, although they also outperformed the margin perceptron with only local features ("re-ranking 2" seems better than "re-ranking 1"). Table 7 shows the training time of each algorithm.[12] Our algorithm is much faster than the reranking approach that uses cross-validation training, while achieving the same or higher level of performance.

7 Discussion

As we mentioned, there are some algorithms similar to ours (Collins and Roark, 2004; Daumé III and Marcu, 2005; McDonald and Pereira, 2006; Liang et al., 2006). The differences of our algorithm from these algorithms are as follows.

Daumé III and Marcu (2005) presented the method called LaSO (Learning as Search Optimization), in which intractable exact inference is approximated by optimizing the behavior of the search process. The method can access non-local features at each search point, if their values can be determined from the search decisions already made. They provided robust training algorithms with guaranteed convergence for this framework. However, a difference is that our method can use non-local features whose value depends on all labels throughout training, and it is unclear whether the features whose values can only be determined at the end of the search (e.g., majority features) can be learned effectively with such an incremental manner of LaSO.

The algorithm proposed by McDonald and Pereira (2006) is also similar to ours. Their target was non-projective dependency parsing, where exact inference is intractable. Instead of using

[11] The weights were re-trained for the second model.

[12] Training time was measured on a machine with 2.33 GHz QuadCore Intel Xeons and 8 GB of memory. C was fixed to 5657.

n-best/re-scoring approach as ours, their method modifies the single best projective parse, which can be found efficiently, to find a candidate with higher score under non-local features. Liang et al. (2006) used n candidates of a beam search in the Collins' perceptron algorithm for machine translation. Collins and Roark (2004) proposed an approximate incremental method for parsing. Their method can be used for sequence labeling as well. These studies, however, did not explain the validity of their updating methods in terms of convergence.

To achieve robust training, Daumé III and Marcu (2005) employed the averaged perceptron (Collins, 2002a) and ALMA (Gentile, 2001). Collins and Roark (2004) used the averaged perceptron (Collins, 2002a). McDonald and Pereira (2006) used MIRA (Crammer et al., 2006). On the other hand, we employed the margin perceptron (Krauth and Mézard, 1987), extending it to sequence labeling. We demonstrated that this greatly improved robustness.

With regard to the local update, (B), in Algorithm 4.2, "early updates" (Collins and Roark, 2004) and "y-good" requirement in (Daumé III and Marcu, 2005) resemble our local update in that they tried to avoid the situation where the correct answer cannot be output. Considering such commonality, the way of combining the local update and the non-local update might be one important key for further improvement.

It is still open whether these differences are advantages or disadvantages. However, we think our algorithm can be a contribution to the study for incorporating non-local features. The convergence guarantee is important for the confidence in the training results, although it does not mean high performance directly. Our algorithm could at least improve the accuracy of NER with non-local features and it was indicated that our algorithm was superior to the re-ranking approach in terms of accuracy and training cost. However, the achieved accuracy was not better than that of related work (Finkel et al., 2005; Krishnan and Manning, 2006) based on CRFs. Although this might indicate the limitation of perceptron-based methods, it has also been shown that there is still room for improvement in perceptron-based algorithms as our margin perceptron algorithm demonstrated.

8 Conclusion

In this paper, we presented a new perceptron algorithm for learning with non-local features. We think the proposed algorithm is an important step towards achieving our final objective. We would like to investigate various types of new non-local features using the proposed algorithm in future work.

Appendix A: Convergence of Algorithm 4.2

Let $\boldsymbol{\alpha}^k$ be a weight vector before the kth update and ϵ_k be a variable that takes 1 when the kth update is done in (A) and 0 when done in (B). The update rule can then be written as $\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k + \epsilon_k(\Phi^{a*} - \Phi^a + (1-\epsilon_k)(\Phi^{l*} - \Phi^l)$.[13] First, we obtain

$$\begin{aligned}\boldsymbol{\alpha}^{k+1}\cdot\boldsymbol{U}^l &= \boldsymbol{\alpha}^k\cdot\boldsymbol{U}^l + \epsilon_k(\Phi^{a*}\cdot\boldsymbol{U}^l - \Phi^a\cdot\boldsymbol{U}^l)\\ &\quad +(1-\epsilon_k)(\Phi^{l*}\cdot\boldsymbol{U}^l - \Phi^l\cdot\boldsymbol{U}^l)\\ &\geq \boldsymbol{\alpha}^k\cdot\boldsymbol{U}^l + \epsilon_k\delta + (1-\epsilon_k)\delta\\ &= \boldsymbol{\alpha}^k\cdot\boldsymbol{U}^l + \delta \geq \boldsymbol{\alpha}^1\cdot\boldsymbol{U}^l + k\delta = k\delta\end{aligned}$$

Therefore, $(k\delta)^2 \leq (\boldsymbol{\alpha}^{k+1}\cdot\boldsymbol{U}^l)^2 \leq (||\boldsymbol{\alpha}^{k+1}||||\boldsymbol{U}^l||)^2 = ||\boldsymbol{\alpha}^{k+1}||^2$ — (1). On the other hand, we also obtain

$$\begin{aligned}||\boldsymbol{\alpha}^{k+1}||^2 &\leq ||\boldsymbol{\alpha}^k||^2 + 2\epsilon_k\boldsymbol{\alpha}^k(\Phi^{a*} - \Phi^a)\\ &\quad +2(1-\epsilon_k)\boldsymbol{\alpha}^k(\Phi^{l*} - \Phi^l)\\ &\quad +\{\epsilon_k(\Phi^{a*} - \Phi^a) + (1-\epsilon_k)(\Phi^{l*} - \Phi^l)\}^2\\ &\leq ||\boldsymbol{\alpha}^k||^2 + 2C + R^2\\ &\leq ||\boldsymbol{\alpha}^1||^2 + k(R^2+2C) = k(R^2+2C) \text{— (2)}\end{aligned}$$

We used $\boldsymbol{\alpha}^k(\Phi^{a*} - \Phi^a) \leq C^a$, $\boldsymbol{\alpha}^k(\Phi^{l*} - \Phi^l) \leq C^l$ and $C^l = C^a = C$ to derive $2C$ in the second inequality. We used $||\Phi^{l*} - \Phi^l|| \leq ||\Phi^{a*} - \Phi^a|| \leq R$ to derive R^2.

Combining (1) and (2), we obtain $k \leq (R^2 + 2C)/\delta^2$. Substituting this into (2) gives $||\boldsymbol{\alpha}^k|| \leq (R^2+2C)/\delta$. Since $\boldsymbol{y}* = \boldsymbol{y}'$ and $\Phi^{a*}\cdot\boldsymbol{\alpha} - \Phi^{a''}\cdot\boldsymbol{\alpha} > C$ after convergence, we obtain

$$\gamma'(\boldsymbol{\alpha}) = \min_{\boldsymbol{x}_i}\frac{\Phi^{a*}\cdot\boldsymbol{\alpha} - \Phi^{a''}\cdot\boldsymbol{\alpha}}{||\boldsymbol{\alpha}||} \geq C\delta/(2C+R^2).$$

[13] We use the shorthand $\Phi^{a*} = \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$, $\Phi^a = \Phi^a(\boldsymbol{x}_i, \boldsymbol{y})$, $\Phi^{l*} = \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$, and $\Phi^l = \Phi^l(\boldsymbol{x}_i, \boldsymbol{y})$ where $\boldsymbol{y}$ represents the candidate used to update ($\boldsymbol{y}'$, $\boldsymbol{y}''$, $\boldsymbol{y}^1$, or $\boldsymbol{y}^2$).

References

D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

R. Bunescu and R. J. Mooney. 2004. Collective information extraction with relational markov networks. In *ACL 2004*.

S. F. Chen and R. Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL 2002*, pages 263–270.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL 2004*.

M. Collins. 2002a. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*.

M. Collins. 2002b. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *ACL 2002*.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*.

H. Daumé III and D. Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML 2005*.

J. R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL 2005*.

C. Gentile. 2001. A new approximate maximal margin classification algorithm. *JMLR*, 3.

R. Herbrich and T. Graepel. 2000. Large scale Bayes point machines. In *NIPS 2000*.

W. Krauth and M. Mézard. 1987. Learning algorithms with optimal stability in neural networks. *Journal of Physics A 20*, pages 745–752.

V. Krishnan and C. D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognitioin. In *ACL-COLING 2006*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, pages 282–289.

Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola. 2002. The perceptron algorithm with uneven margins. In *ICML 2002*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL-COLING 2006*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL 2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *ACL 2005*.

T. Nakagawa and Y. Matsumoto. 2006. Guessing parts-of-speech of unknown words using global information. In *ACL-COLING 2006*.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *third ACL Workshop on very large corpora*.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psycological Review*, pages 386–407.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *ICML 2005*.

S. Sarawagi and W. W. Cohen. 2004. Semi-Markov random fields for information extraction. In *NIPS 2004*.

L. Shen and A. K. Joshi. 2004. Flexible margin selection for reranking with full pairwise samples. In *IJCNLP 2004*.

F. K. Soong and E. Huang. 1991. A tree-trellis based fast search for finding the n best sentence hypotheses in continuous speech recognition. In *ICASSP-91*.

C. Sutton and A. McCallum. 2004. Collective segmenation and labeling of distant entitites in information extraction. University of Massachusetts Rechnical Report TR 04-49.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *NIPS 2003*.

E. F. Tjong, K. Sang, and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL 2003*.

Extending a Thesaurus in the Pan-Chinese Context

Oi Yee Kwong and Benjamin K. Tsou
Language Information Sciences Research Centre
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong
{rlolivia,rlbtsou}@cityu.edu.hk

Abstract

In this paper, we address a unique problem in Chinese language processing and report on our study on extending a Chinese thesaurus with *region-specific* words, mostly from the financial domain, from various Chinese speech communities. With the larger goal of automatically constructing a Pan-Chinese lexical resource, this work aims at taking an existing semantic classificatory structure as leverage and incorporating new words into it. In particular, it is important to see if the classification could accommodate new words from heterogeneous data sources, and whether simple similarity measures and clustering methods could cope with such variation. We use the cosine function for similarity and test it on automatically classifying 120 target words from four regions, using different datasets for the extraction of feature vectors. The automatic classification results were evaluated against human judgement, and the performance was encouraging, with accuracy reaching over 85% in some cases. Thus while human judgement is not straightforward and it is difficult to create a Pan-Chinese lexicon manually, it is observed that combining simple clustering methods with the appropriate data sources appears to be a promising approach toward its automatic construction.

1 Introduction

Large-scale semantic lexicons are important resources for many natural language processing (NLP) tasks. For a significant world language such as Chinese, it is especially critical to capture the substantial *regional variation* as an important part of the lexical knowledge, which will be useful for many NLP applications, including natural language understanding, information retrieval, and machine translation. Existing Chinese lexical resources, however, are often based on language use in one particular region and thus lack the desired comprehensiveness.

Toward this end, Tsou and Kwong (2006) proposed a comprehensive Pan-Chinese lexical resource, based on a large and unique synchronous Chinese corpus as an authentic source for lexical acquisition and analysis across various Chinese speech communities. To allow maximum *versatility* and *portability*, it is expected to document the core and universal substances of the language on the one hand, and also the more subtle variations found in different communities on the other. Different Chinese speech communities might share lexical items in the same form but with different meanings. For instance, the word 居屋 refers to general housing in Mainland China but specifically to housing under the Home Ownership Scheme in Hong Kong; and while the word 住房 is similar to 居屋 to mean general housing in Mainland China, it is rarely seen in the Hong Kong context.

Hence, the current study aims at taking an existing Chinese thesaurus, namely the *Tongyici Cilin* 同義詞詞林, as leverage and extending it with lexical items specific to individual Chinese speech communities. In particular, the feasibility depends on the following issues: (1) Can lexical items from various Chinese speech communities, that is, from such heterogeneous sources, be classified as effectively with methods shown to work for clustering

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 325–333, Prague, June 2007. ©2007 Association for Computational Linguistics

closely related words from presumably the same, or homogenous, source? (2) Could existing semantic classificatory structures accommodate concepts and expressions specific to individual Chinese speech communities?

Measuring similarity will make sense only if the feature vectors of the two words under comparison are directly comparable. There is usually no problem if both words and their contextual features are from the same data source. Since Tongyici Cilin (or simply Cilin hereafter) is based on the vocabulary used in Mainland China, it is not clear how often these words will be found in data from other places, and even if they are found, how well the feature vectors extracted could reflect the expected usage or sense. Our hypothesis is that it will be more effective to classify new words from Mainland China with respect to Cilin categories, than to do the same on new words from regions outside Mainland China. Furthermore, if this hypothesis holds, one would need to consider separate mechanisms to cluster heterogeneous region-specific words in the Pan-Chinese context.

Thus in the current study we sampled 30 target words specific to each of Beijing, Hong Kong, Singapore, and Taipei, from the financial domain; and used the cosine similarity function to classify them into one or more of the semantic categories in Cilin. The automatic classification results were compared with a simple baseline method, against human judgement as the gold standard. In general, an accuracy of up to 85% could be reached with the top 15 candidates considered. It turns out that our hypothesis is supported by the Taipei test data, whereas the data heterogeneity effect is less obvious in Hong Kong and Singapore test data, though the effect on individual test items varies.

In Section 2, we will briefly review related work and highlight the innovations of the current study. In Sections 3 and 4, we will describe the materials used and the experimental setup respectively. Results will be presented and discussed with future directions in Section 5, followed by a conclusion in Section 6.

2 Related Work

To build a semantic lexicon, one has to identify the relation between words within a semantic hierarchy, and to group similar words together into a class. Previous work on automatic methods for building semantic lexicons could be divided into two main groups. One is automatic thesaurus acquisition, that is, to identify synonyms or topically related words from corpora based on various measures of similarity (e.g. Riloff and Shepherd, 1997; Thelen and Riloff, 2002). For instance, Lin (1998) used dependency relation as word features to compute word similarities from large corpora, and compared the thesaurus created in such a way with WordNet and Roget classes. Caraballo (1999) selected head nouns from conjunctions and appositives in noun phrases, and used the cosine similarity measure with a bottom-up clustering technique to construct a noun hierarchy from text. Curran and Moens (2002) explored a new similarity measure for automatic thesaurus extraction which better compromises with the speed/performance tradeoff. You and Chen (2006) used a feature clustering method to create a thesaurus from a Chinese newspaper corpus.

Another line of research, which is more closely related with the current study, is to extend existing thesauri by classifying new words with respect to their given structures (e.g. Tokunaga *et al.*, 1997; Pekar, 2004). An early effort along this line is Hearst (1992), who attempted to identify hyponyms from large text corpora, based on a set of lexico-syntactic patterns, to augment and critique the content of WordNet. Ciaramita (2002) compared several models in classifying nouns with respect to a simplified version of WordNet and signified the gain in performance with morphological features. For Chinese, Tseng (2003) proposed a method based on morphological similarity to assign a Cilin category to unknown words from the Sinica corpus which were not in the Chinese Electronic Dictionary and Cilin; but somehow the test data were taken from Cilin, and therefore could not really demonstrate the effectiveness with unknown words found in the Sinica corpus.

The current work attempts to classify new words with an existing thesaural classificatory structure. However, the usual practice in past studies is to test with a portion of data from the thesaurus itself and evaluate the results against the original classification of those words. This study is thus different in the following ways: (1) The test data (i.e. the target words to be classified) were not taken from the thesaurus, but extracted from corpora and these words were unknown to the thesaurus. (2) The target words were not limited to nouns. (3) Automatic classification results were compared with a baseline method and with the manual judgement of several linguistics students constituting the gold standard. (4) In view of the heterogeneous nature of the Pan-Chinese context, we experimented with extracting feature vectors from different datasets.

3 Materials

3.1 The Tongyici Cilin

The Tongyici Cilin (同義詞詞林) (Mei *et al.*, 1984) is a Chinese synonym dictionary, or more often known as a Chinese thesaurus in the tradition of the Roget's Thesaurus for English. The Roget's Thesaurus has about 1,000 numbered semantic heads, more generally grouped under higher level semantic classes and subclasses, and more specifically differentiated into paragraphs and semicolon-separated word groups. Similarly, some 70,000 Chinese lexical items are organized into a hierarchy of broad conceptual categories in Cilin. Its classification consists of 12 top-level semantic classes, 94 subclasses, 1,428 semantic heads and 3,925 paragraphs. It was first published in the 1980s and was based on lexical usages mostly of post-1949 Mainland China. The Appendix shows some example subclasses. In the following discussion, we will mainly refer to the subclass level and semantic head level.

3.2 The LIVAC Synchronous Corpus

LIVAC (http://www.livac.org) stands for Linguistic Variation in Chinese Speech Communities. It is a synchronous corpus developed and dynamically maintained by the Language Information Sciences Research Centre of the City University of Hong Kong since 1995 (Tsou and Lai, 2003). The corpus consists of newspaper articles collected regularly and synchronously from six Chinese speech communities, namely Hong Kong, Beijing, Taipei, Singapore, Shanghai, and Macau. Texts collected cover a variety of domains, including front page news stories, local news, international news, editorials, sports news, entertainment news, and financial news. Up to December 2006, the corpus has already accumulated over 200 million character tokens which, upon automatic word segmentation and manual verification, amount to over 1.2 million word types.

For the present study, we made use of the subcorpora collected over the 9-year period 1995-2004 from Beijing (BJ), Hong Kong (HK), Singapore (SG), and Taipei (TW). In particular, we made use of the *financial news* sections in these subcorpora, from which we extracted feature vectors for comparing similarity between a given target word and a thesaurus class, which is further explained in Section 4.3. Table 1 shows the sizes of the subcorpora.

3.3 Test Data

Instead of using a portion of Cilin as the test data, we extracted unique lexical items from the various subcorpora above, and classified them with respect to the Cilin classification.

Kwong and Tsou (2006) observed that among the unique lexical items found from the individual subcorpora, only about 30-40% are covered by Cilin, but not necessarily in the expected senses. In other words, Cilin could in fact be enriched with over 60% of the unique items from various regions.

In the current study, we sampled the most frequent 30 words from each of these unique item lists for testing. Classification was based on their similarity with each of the Cilin subclasses, compared by the cosine measure, as discussed in Section 4.3.

Subcorpus	**Size of Financial News Sections** (rounded to nearest 1K)	
	Word Token	**Word Type**
BJ	232K	20K
HK	970K	38K
SG	621K	28K
TW	254K	22K

Table 1 Sizes of Individual Subcorpora

4 Experiments

4.1 Human Judgement

Three undergraduate linguistics students and one research student on computational linguistics from the City University of Hong Kong were asked to do the task. The undergraduate students were raised in Hong Kong and the research student in Mainland China. They were asked to assign what they consider the most appropriate Cilin category (up to the semantic head level, i.e. third level in the

Cilin structure) to each of the 120 target words. The inter-annotator agreement was measured by the *Kappa* statistic (Siegel and Castellan, 1988), at both the subclass and semantic head levels. Results on the human judgement are discussed in Section 5.1.

4.2 Creating Gold Standard

The "gold standard" was set at both the subclass level and semantic head level. For each level, we formed a "strict" standard for which we considered all categories assigned by at least two judges to a word; and a "loose" standard for which we considered all categories assigned by one or more judges. For evaluating the automatic classification in this study, however, we only experimented with the loose standard at the subclass level.

4.3 Automatic Classification

Each target word was automatically classified with respect to the Cilin subclasses based on the similarity between the target word and each subclass.

We compute the similarity by the cosine between the two corresponding feature vectors. The feature vector of a given target word contains *all its co-occurring content words* in the corpus within a window of ±5 words (excluding many general adjectives and adverbs, and numbers and proper names were all ignored). The feature vector of a Cilin subclass is based on the union of the features (i.e. co-occurring words in the corpus) from all individual members in the subclass.

The cosine of two feature vectors is computed as

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|}$$

In view of the difference in the feature space of a target word and a whole class of words, and thus the potential difference in the number of occurrence of individual features, we experimented with two versions of the cosine measurement, namely binary vectors and real-valued vectors.

In addition, as mentioned in previous sections, we also experimented with the following conditions: whether feature vectors for the Cilin subclasses were extracted from the subcorpus where a given target word originates, or from the Beijing subcorpus which is assumed to be representative of language use in Mainland China. All automatic classification results were evaluated against the gold standard based on human judgement.

4.4 Baseline

To evaluate the effectiveness of the automatic classification, we adopted a simple baseline measure by ranking the 94 subclasses in descending order of the number of words they cover. In other words, assuming the bigger the subclass size, the more likely it covers a new term, thus we compared the top-ranking subclasses with the classifications obtained from the automatic method using the cosine measure.

5 Results and Discussion

5.1 Response from Human Judges

All human judges reported difficulties in various degrees in assigning Cilin categories to the target words. The major problem comes from the regional specificity and thus the unfamiliarity of the judges with the respective lexical items and contexts. For instance, students grown up in Hong Kong were most familiar with the Hong Kong data, and slightly less so with the Beijing data, but more often had the least ideas for the Taipei and Singapore data. The research student from Mainland China had no problem with Beijing data and the lexical items in Cilin, but had a hard time figuring out the meaning for words from Hong Kong, Taipei and Singapore. For example, all judges reported problem with the term 自撮, one of the target words from Singapore referring to 自撮股市 (CLOB in the Singaporean stock market), which is really specific to Singapore.

The demand on cross-cultural knowledge thus poses a challenge for building a Pan-Chinese lexical resource manually. Cilin, for instance, is quite biased in language use in Mainland China, and it requires experts with knowledge of a wide variety of Chinese terms to be able to manually classify lexical items specific to other Chinese speech communities. It is therefore even more important to devise robust ways for automatic acquisition of such a resource.

Notwithstanding the difficulty, the inter-annotator agreement was quite satisfactory. At the subclass level, we found *K*=0.6870. At the semantic head level, we found *K*=0.5971. Both figures are statistically significant.

5.2 Gold Standard

As mentioned, we set up a loose standard and a strict standard at both the subclass and semantic head level. In general, the judges managed to reach some consensus in all cases, except for two words from Singapore. For these two cases, we considered all categories assigned by any of the judges for both standards.

The gold standards were verified by the authors. Although in several cases the judges did not reach complete agreement with one another, we found that their decisions reflected various possible perspectives to classify a given word with respect to the Cilin classification; and the judges' assignments, albeit varied, were nevertheless reasonable in one way or another.

5.3 Evaluating Automatic Classification

In the following discussion, we will refer to the various testing conditions for each group of target words with labels in the form of Cos-<Vector Type>-<Target Words>-<Cilin Feature Source>. Thus the label Cos-Bin-hk-hk means testing on Hong Kong target words with binary vectors and extracting features for the Cilin words from the Hong Kong subcorpus; and the label Cos-RV-sg-bj means testing on Singapore target words with real-valued vectors and extracting features for the Cilin words from the Beijing subcorpus. For each target word, we evaluated the automatic classification (and the baseline ranking) by matching the human decisions with the top N candidates. If any of the categories suggested by the human judges is covered, the automatic classification is considered accurate. The results are shown in Figure 1 for test data from individual regions.

Overall speaking, the results are very encouraging, especially in view of the number of categories (over 90) we have at the subclass level. An accuracy of 80% or more is obtained in general if the top 15 candidates were considered, which is much higher than the baseline result in all cases. Table 2 shows some examples with appropriate classification within the Top 3 candidates. The two-letter codes in the "Top 3" column in Table 2 refer to the subclass labels, and the code in bold is the one matching human judgement.

In terms of the difference between binary vectors and real-valued vectors in the similarity measurement, the latter almost always gave better results. This was not surprising as we expected by using real-valued vectors we could be less affected by the potential huge difference in the feature space and the number of occurrence of the features for a Cilin subclass and a target word.

As for extracting features for Cilin subclasses from the Beijing subcorpus or other subcorpora, the difference is more obvious for the Singapore and Taipei target words. We will discuss the results for each group of target words in detail below.

5.4 Performance on Individual Sources

Target words from Beijing were expected to have a relatively higher accuracy because they are homogenous with the Cilin content. It turned out, however, the accuracy only reached 73% with top 15 candidates and 83% with top 20 candidates even under the Cos-RV-bj-bj condition. Words like 非典 (SARS), 節水 (save water), 產業化 (industrialize / industrialization), 合格率 (passing rate) and 傳銷 (multi-level marketing) could not be successfully classified.

Results were surprisingly good for target words from the Hong Kong subcorpus. Under the Cos-RV-hk-hk condition, the accuracy was 87% with top 15 candidates and even over 95% with top 20 candidates considered. Apart from this high accuracy, another unexpected observation is the lack of significant difference between Cos-RV-hk-hk and Cos-RV-hk-bj. One possible reason is that the relatively larger size of the Hong Kong subcorpus might have allowed enough features to be extracted even for the Cilin words. Nevertheless, the similar results from the two conditions might also suggest that the context in which Cilin words are used might be relatively similar in the Hong Kong subcorpus and the Beijing subcorpus, as compared with other communities.

Similar trends were observed from the Singapore target words. Looking at Cos-RV-sg-sg and Cos-RV-sg-bj, it appears that extracting feature vectors for the Cilin words from the Singapore subcorpus leads to better performance than extracting them from the Beijing subcorpus. It suggests that although the Singapore subcorpus shares those words in Cilin, the context in which they are used might be slightly different from their use in Mainland China. Thus extracting their contextual features from the Singapore subcorpus might better reflect their usage and makes it more comparable with the unique target words from Singapore. Such possible difference in contextual features with shared lexical items between different Chinese speech communities would require further investigation, and will form part of our future work as discussed below. Despite the above observation from the accuracy figures, the actual effect, however, seems to vary on individual lexical items. Table 3 shows some examples of target words which received similar (with white cells) and very different (with shaded cells) classification respectively under the two conditions. It appears that the region-specific but common concepts like 寫字樓 (office), 組屋 (apartment), 私宅 (private residence), which relate to building or housing, were affected most.

Taipei data, on the contrary, seems to be more affected by the different testing conditions. Cos-Bin-tw-bj and Cos-RV-tw-bj produced similar results, and both conditions showed better results than Cos-RV-tw-tw. This supports our hypothesis that the effect of data heterogeneity is so apparent that it is much harder to classify target words unique to Taipei with respect to the Cilin categories. In addition, as Kwong and Tsou (2006) observed, Beijing and Taipei data share the least number of lexical items, among the four regions under investigation. Hence, words in Cilin might not have the appropriate contextual feature vectors extracted from the Taipei subcorpus.

The different results for individual regions might be partly due to the endocentric and exocentric nature of influence in lexical innovation (e.g. Tsou, 2001) especially with respect to the financial domain and the history of capitalism in individual regions. This factor is worth further investigation.

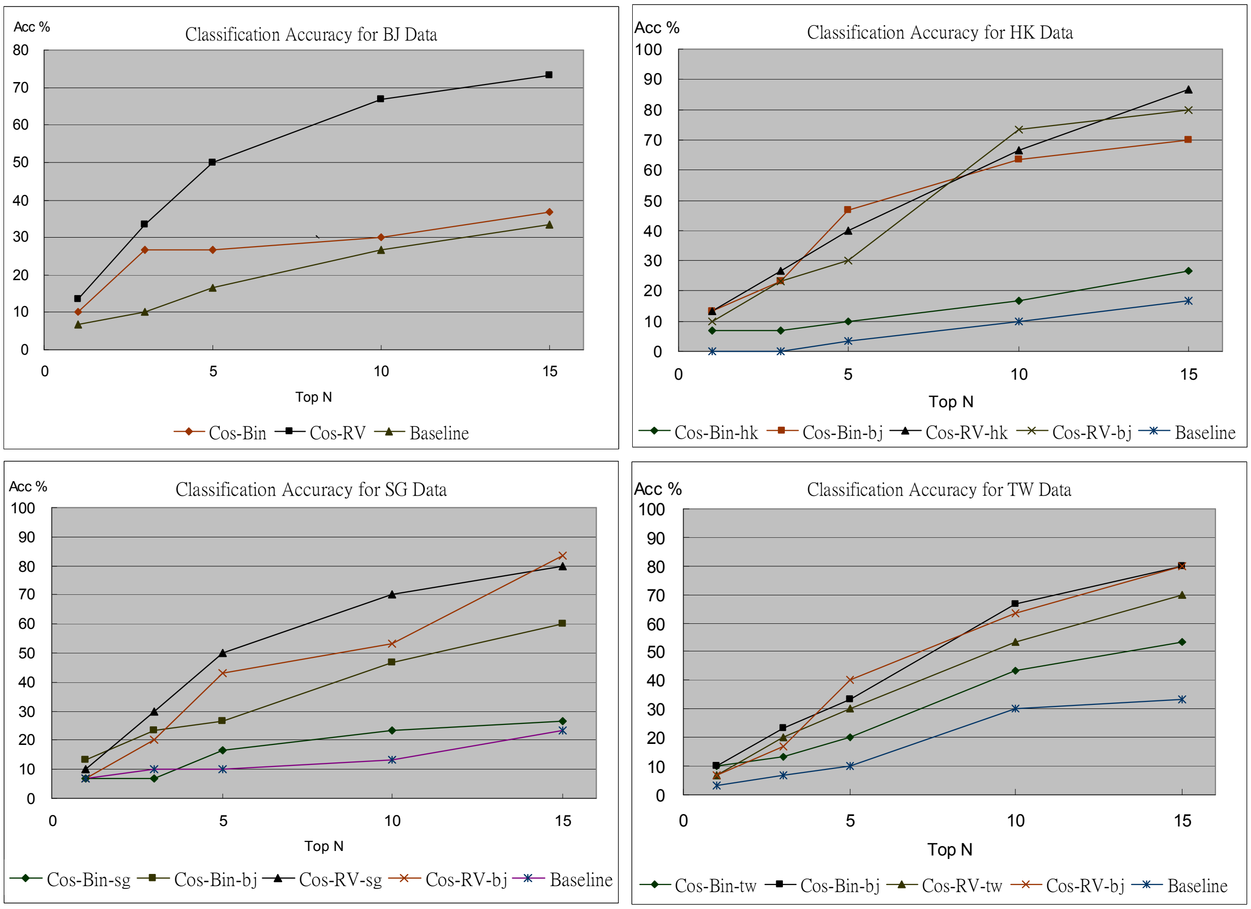


Figure 1 Classification Results with Top N Candidates

No.	Region	Word	Top 3
1	BJ	退耕還林	**Di** Gb Df
2	BJ	面料	**Bq** Ae Hd
3	BJ	煤礦	**Bm** Hi Hd
4	BJ	抓好	**Hj** Di Hd
5	BJ	下崗	Aa **If** Ae
6	HK	銷情	**Da** Cb Bi
7	HK	寬頻	**Bb** Jc Hi
8	HK	紅籌	**Dj** Da Hi
9	HK	息率	Bi **Dj** Dn
10	HK	地產股	Bi **Dj** Gb
11	SG	財年	**Ca** Dm Hi
12	SG	賣空	**Ig** He Dj
13	SG	獻議	Dm Dj **Hi**
14	SG	脫售	Dm Dj **He**
15	SG	准將	Hi Hg **Af**
16	TW	金控	**Dm** Hd Hi
17	TW	個股	Jb Dn **Dj**
18	TW	房市	Ja Ca **He**
19	TW	現金卡	Hf **Dj** Dm
20	TW	存底	**Dj** Ed Ca

Table 2 Examples of Correct Classification (Top 3)[1]

5.5 General Discussions and Future Work

As mentioned in a previous section, the test data in this study were not taken from the thesaurus itself, but were unknown words to the thesaurus. They were extracted from corpora, and were not limited to nouns. We found in this study that the simple cosine measure, which used to be applied for clustering contextually similar words from homogenous sources, performs quite well in general for classifying these unseen words with respect to the Cilin subclasses. The automatic classification results were compared with the manual judgement of several linguistics students. In addition to providing a gold standard for evaluating the automatic classification results in this study, the human judgement on the one hand proves that the Cilin classificatory structure could accommodate region-specific lexical items; but on the other hand also suggests how difficult it would be to construct such a Pan-Chinese lexicon manually as rich cultural and linguistic knowledge would be required. Moreover, we started with Cilin as the established semantic classification and attempted to classify words specific to Beijing, Hong Kong, Singapore, and Taipei respectively. The heterogeneity of sources did not seem to hamper the similarity measure on the whole, provided appropriate datasets are used for feature extraction, although the actual effect seemed to vary on individual lexical items.

No.	Source	Word	Ranking of 1st appropriate class	
			Cos-RV-hk-hk, etc.	Cos-RV-hk-bj, etc.
1	HK	銷情	1	1
2	HK	寬頻	1	1
3	HK	紅籌	1	1
4	HK	加推	2	10
5	HK	低位	19	5
6	HK	寫字樓	13	30
7	SG	財政年	2	2
8	SG	賣空	2	1
9	SG	附加股	5	4
10	SG	組屋	1	12
11	SG	容積率	1	9
12	SG	私宅	8	26
13	TW	存底	1	1
14	TW	個股	4	3
15	TW	金控	5	1
16	TW	投信	18	4
17	TW	成長率	12	5
18	TW	現金卡	8	2

Table 3 Different Impact on Individual Items[2]

Despite the encouraging results with the top 15 candidates in the current study, it is desirable to improve the accuracy for the system to be useful in

[1] English gloss: 1-restoring agricultural lands for afforestation, 2-material, 3-coal mine, 4-to seize (an opportunity), 5-unemployed, 6-sales performance, 7-broadband, 8-red chip, 9-interest rate, 10-property stocks, 11-financial year, 12-sell short, 13-proposal, 14-sell, 15-brigadier general, 16-financial holdings, 17-individual stocks, 18-property market, 19-cash card, 20-stub.

[2] English gloss: 1-sales performance, 2-broadband, 3-red chip, 4-add (supply to market), 5-low level, 6-office, 7-financial year, 8-sell short, 9-rights issue, 10-apartment, 11-holding space rate, 12-private residence, 13-stub, 14-individual stocks, 15-financial holdings, 16-investment trust, 17-growth rate, 18-cash card.

practice. Hence our next step is to expand the test data size and to explore alternative methods such as using a nearest neighbour approach. In addition, we plan to further the investigation in the following directions. First, we will experiment with the automatic classification at the Cilin semantic head level, which is much more fine-grained than the subclasses. The fine-grainedness might make the task more difficult, but at the same time the more specialized grouping might pose less ambiguity for classification. Second, we will further experiment with classifying words from other special domains like sports, as well as the general domain. Third, we will study the classification in terms of the part-of-speech of the target words, and their respective requirements on the kinds of features which give best classification performance.

The current study only dealt with presumably Modern Standard Chinese in different communities, and it could potentially be expanded to handle various dialects within a common resource, eventually benefiting speech lexicons and applications at large.

6 Conclusion

In this paper, we have reported our study on a unique problem in Chinese language processing, namely extending a Chinese thesaurus with new words from various Chinese speech communities, including Beijing, Hong Kong, Singapore and Taipei. The critical issues include whether the existing classificatory structure could accommodate concepts and expressions specific to various Chinese speech communities, and whether the difference in textual sources might pose difficulty in using conventional similarity measures for the automatic classification. Our experiments, using the cosine function to measure similarity and testing with various sources for extracting contextual vectors, suggest that the classification performance might depend on the compatibility between the words in the thesaurus and the sources from which the target words are drawn. Evaluated against human judgement, an accuracy of over 85% was reached in some cases, which were much higher than the baseline and were very encouraging in general. While human judgement is not straightforward and it is difficult to create a Pan-Chinese lexicon manually, combining simple classification methods with the appropriate data sources seems to be a promising approach toward its automatic construction.

Acknowledgements

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 1317/03H).

Appendix

The following table shows some examples of the Cilin subclasses:

Class	Subclasses
A 人 (Human)	Aa … Ae 職業 (Occupation) Af 身份 (Identity) … An
B 物 (Things)	Ba … Bb 擬狀物 (Shape) … Bi 動物 (Animal)… Bm 材料 (Material)…Bq 衣物 (Clothing) … Br
C 時間與空間 (Time and Space)	Ca 時間 (Time) Cb 空間 (Space)
D 抽象事物 (Abstract entities)	Da 事情 情況 (Condition) … Df 意識 (Ideology) … Di 社會 政法 (Society) Dj 經濟 (Economics) … Dm 機構 (Organization) Dn 數量單位 (Quantity)
E 特徵 (Characteristics)	Ea … Ed 性質 (Property)… Ef
F 動作 (Action)	Fa … Fd
G 心理活動 (Psychological activities)	Ga … Gb 心理活動 (Psychological activities)… Gc
H 活動 (Activities)	Ha … He 經濟活動 (Economic activities) … Hd 生產 (Production) … Hf 交通運輸 (Transportation) Hg 教衛科研 (Scientific research)… Hi 社交 (Social contact) Hj 生活 (Livelihood)
I 現象與狀態 (Phenomenon and state)	Ia … If 境遇 (Circumstance) Ig 始末 (Process)… Ih
J 關聯 (Association)	Ja 聯繫 (Liaison) Jb 異同 (Similarity and Difference) Jc 配合 (Matching) … Je
K 助語 (Auxiliary words)	Ka … Kf
L 敬語 (Respectful expressions)	

References

Caraballo, S.A. (1999) Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, Maryland, USA, pp.120-126.

Ciaramita, M. (2002) Boosting automatic lexical acquisition with morphological information. In *Proceedings of the ACL'02 Workshop on Unsupervised Lexical Acquisition*, Philadelphia, USA, pp.17-25.

Curran, J.R. and Moens, M. (2002) Improvements in Automatic Thesaurus Extraction. In *Proceedings of the ACL'02 Workshop on Unsupervised Lexical Acquisition*, Philadelphia, USA, pp.59-66.

Hearst, M. (1992) Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, Nantes, France, pp.539-545.

Kwong, O.Y. and Tsou, B.K. (2006) Feasibility of Enriching a Chinese Synonym Dictionary with a Synchronous Chinese Corpus. In T. Salakoski, F. Ginter, S. Pyysalo and T. Pahikkala (Eds.), *Advances in Natural Language Processing: Proceedings of FinTAL 2006.* Lecture Notes in Artificial Intelligence, Vol.4139, pp.322-332, Springer-Verlag.

Lin, D. (1998) Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, Montreal, Canada, pp.768-774.

Mei *et al.* 梅家駒、竺一鳴、高蘊琦、殷鴻翔 (1984) 《同義詞詞林》 (*Tongyici Cilin*). 商務印書館 (Commerical Press) / 上海辭書出版社.

Pekar, V. (2004) Linguistic Preprocessing for Distributional Classification of Words. In *Proceedings of the COLING2004 Workshop on Enhancing and Using Electronic Dictionaries*, Geneva.

Riloff, E. and Shepherd, J. (1997) A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Providence, Rhode Island, pp.117-124.

Siegel, S. and Castellan, N.J. (1988) *Nonparametric Statistics for the Behavioral Sciences (2nd Ed.)*. McGraw-Hill.

Thelen, M. and Riloff, E. (2002) A Bootstrapping Method for Learning Semantic Lexicons using Extraction Pattern Contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, Philadelphia, USA.

Tokunaga, T., Fujii, A., Iwayama, M., Sakurai, N. and Tanaka, H. (1997) Extending a thesaurus by classifying words. In *Proceedings of the ACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, Madrid, pp.16-21.

Tseng, H. (2003) Semantic Classification of Chinese Unknown Words. In the *Proceedings of the ACL-2003 Student Research Workshop, Companion Volume to the Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Tsou, B.K. (2001) Language Contact and Lexical Innovation. In M. Lackner, I. Amelung and J. Kurtz (Eds.), *New Terms for New Ideas: Western Knowledge and Lexical Change in Late Imperial China*. Berlin: Brill.

Tsou, B.K. and Kwong, O.Y. (2006) Toward a Pan-Chinese Thesaurus. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Tsou, B.K. and Lai, T.B.Y. 鄒嘉彥、黎邦洋 (2003) 漢語共時語料庫與信息開發. In B. Xu, M. Sun and G. Jin 徐波、孫茂松、靳光瑾 (Eds.), 《中文信息處理若干重要問題》 (*Issues in Chinese Language Processing*). 北京：科學出版社, pp.147-165

You, J-M. and Chen, K-J. (2006) Improving Context Vector Models by Feature Clustering for Automatic Thesaurus Construction. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, COLING-ACL 2006, Sydney, Australia, pp.1-8.

Low-Quality Product Review Detection in Opinion Summarization

Jingjing Liu
Nankai University
Tianjin, China
v-jingil@microsoft.com

Yunbo Cao
Microsoft Research Asia
Beijing, China
yucao@microsoft.com

Chin-Yew Lin
Microsoft Research Asia
Beijing, China
cyl@microsoft.com

Yalou Huang
Nankai University
Tianjin, China
huangyl@nankai.edu.cn

Ming Zhou
Microsoft Research Asia
Beijing, China
mingzhou@microsoft.com

Abstract

Product reviews posted at online shopping sites vary greatly in quality. This paper addresses the problem of detecting low-quality product reviews. Three types of biases in the existing evaluation standard of product reviews are discovered. To assess the quality of product reviews, a set of specifications for judging the quality of reviews is first defined. A classification-based approach is proposed to detect the low-quality reviews. We apply the proposed approach to enhance opinion summarization in a two-stage framework. Experimental results show that the proposed approach effectively (1) discriminates low-quality reviews from high-quality ones and (2) enhances the task of opinion summarization by detecting and filtering low-quality reviews.

1 Introduction

In the past few years, there has been an increasing interest in mining opinions from product reviews (Pang, et al, 2002; Liu, et al, 2004; Popescu and Etzioni, 2005). However, due to the lack of editorial and quality control, reviews on products vary greatly in quality. Thus, it is crucial to have a mechanism capable of assessing the quality of reviews and detecting low-quality/noisy reviews.

Some shopping sites already provide a function of assessing the quality of reviews. For example, Amazon[1] allows users to vote for the helpfulness of each review and then ranks the reviews based on the accumulated votes. However, according to our survey in Section 3, users' votes at Amazon have three kinds of biases as follows: (1) *imbalance vote bias,* (2) *winner circle bias, and* (3) *early bird bias.* Existing studies (Kim et al, 2006; Zhang and Varadarajan, 2006) used these users' votes for training ranking models to assess the quality of reviews, which therefore are subject to these biases.

In this paper, we demonstrate the aforementioned biases and define a standard specification to measure the quality of product reviews. We then manually annotate a set of ground-truth with real world product review data conforming to the specification.

To automatically detect low-quality product reviews, we propose a classification-based approach learned from the annotated ground-truth. The proposed approach explores three aspects of product reviews, namely informativeness, readability, and subjectiveness.

We apply the proposed approach to opinion summarization, a typical opinion mining task. The proposed approach enhances the existing work in a two-stage framework, where the low-quality review detection is applied right before the summarization stage.

Experimental results show that the proposed approach can discriminate low-quality reviews from high-quality ones effectively. In addition, the task of opinion summarization can be enhanced by detecting and filtering low-quality reviews.

[1] http://www.amazon.com

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 334–342, Prague, June 2007. ©2007 Association for Computational Linguistics

The rest of the paper is organized as follows: Section 2 introduces the related work. In Section 3, we define the quality of product reviews. In Section 4, we present our approach to detecting low-quality reviews. In Section 5, we empirically verify the effectiveness of the proposed approach and its use for opinion summarization. Section 6 summarizes our work in this paper and points out the future work.

2 Related Work

2.1 Evaluating Helpfulness of Reviews

The problem of *evaluating helpfulness of reviews* (Kim et al, 2006), also known as *learning utility of reviews* (Zhang and Varadarajan, 2006), is quite similar to our problem of *assessing the quality of reviews*.

In practice, researchers in this area considered the problem as a ranking problem and solved it with regression models. In the process of model training and testing, they used the ground-truth derived from users' votes of helpfulness provided by Amazon. As we will show later in Section 3, these models all suffered from three types of voting bias.

In our work, we avoid using users' votes by developing a specification on the quality of reviews and building a ground-truth according to the specification.

2.2 Mining Opinions from Reviews

One area of research on opinion mining from product reviews is to judge whether a review expresses a positive or a negative opinion. For example, Turney (2006) presented a simple unsupervised learning algorithm in judging reviews as “thumbs up” (recommended) or “thumbs down” (not recommended). Pang et al (2002) considered the same problem and presented a set of supervised machine learning approaches to it. For other work see also Dave et al. (2003), Pang and Lee (2004, 2005).

Another area of research on opinion mining is to extract and summarize users' opinions from product reviews (Hu and Liu, 2004; Liu et al., 2005; Popescu and Etzioni, 2005). Typically, a sentence or a text segment in the reviews is treated as the basic unit. The polarity of users' sentiments on a product feature in each unit is extracted. Then the aggregation of the polarities of individual sentiments is presented to users so that they can have an at-a-glance view on how other experienced users rated on a certain product. The major weakness in the existing studies is that all the reviews, including low-quality ones, are taken into consideration and treated equally for generating the summary. In this paper, we enhance the application by detecting and filtering low-quality reviews. In order to achieve that, we first define what the quality of reviews is.

3 Quality of Product Reviews

In this section, we will first show three biases of users' votes observed on Amazon, and then present our specification on the *quality of product reviews*.

3.1 Amazon Ground-truth

In our study, we use the product reviews on digital cameras crawled from Amazon as our data set. The data set consists of 23,141 reviews on 946 digital cameras. At the Amazon site, users could vote for a review with a “helpful” or “unhelpful” label. Thus, for each review there are two numbers indicating the statistics of these two labels, namely the number of “helpful” votes and that of “unhelpful” ones. Kim et al (2006) used the percentage of “helpful” votes as the measure of evaluating the “quality of reviews” in their experiments. We call the ground-truth based on this measure as “Amazon ground-truth”.

Certainly, the ground-truth has the advantage of convenience. However, we identify three types of biases that make the Amazon ground-truth not always suitable for determining the quality of reviews. We describe these biases in details in the rest of this section.

3.1.1 Imbalance Vote Bias

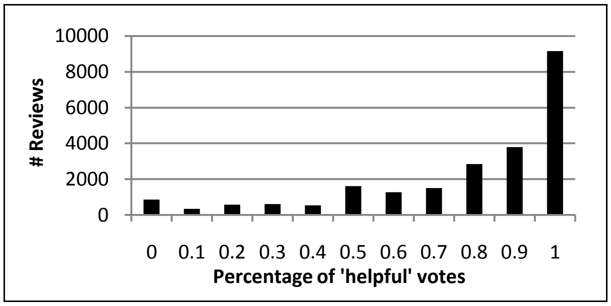


Figure 1. Reviews' percentage scores

At the Amazon site, users tend to value others' opinions positively rather than negatively. From Figure 1, we can see that a half of the 23,141 reviews (corresponding to the two bars on the right of the figure) have more than 90% "helpful" votes, including 9,100 reviews with 100% "helpful" votes. From an in-depth investigation on these highly-voted reviews, we observed that some did not really have as good quality as the votes hint. For example, in Figure 2, the review about *Canon PowerShot S500* receives 40 "helpful" votes out of 40 votes although it only gives very brief description on the product features in its second paragraph. We call this type of bias "imbalance vote" bias.

> *This is my second Canon digital elph camera. Both were great cameras. Recently upgraded to the S500. About 6 months later I get the dreaded E18 error. I searched the Internet and found numerous people having problems. When I determined the problem to be the lens not fully extending I decided to give it a tug. It clicked and the camera came on, ready to take pictures. Turning it off and on produced the E18 again. While turning it on I gave it a nice little bump on the side (where the USB connector is) and the lens popped out on its own. No problems since.*
>
> *It's a nice compact and light camera and takes great photos and videos. Only complaint (other than E18) is the limit of 30-second videos on 640x480 mode. I've got a 512MB compact flash card, I should be able to take as much footage as I have memory in one take.*

Figure 2. An example review

3.1.2 Winner Circle Bias

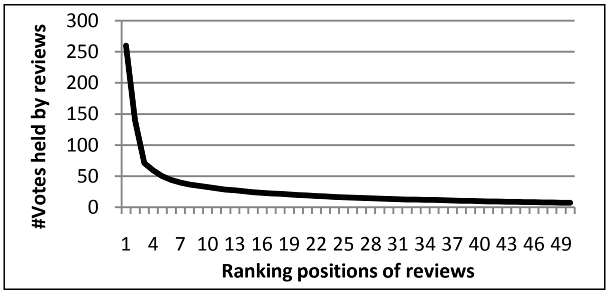


Figure 3. Votes of the top-50 ranked reviews

There also exists a bootstrapping effect of "hot" reviews at the Amazon site. Figure 3 shows the "helpful" votes for the top 50 ranked reviews. The numbers are averaged over 127 digital cameras which have no less than 50 reviews. As shown in this figure, the top two reviews hold more than 250 and 140 votes respectively on average; while the numbers of votes held by lower-ranked reviews decrease exponentially. This is so-called the "winner circle" bias: the more votes a review gains, the more default authority it would appear to the readers, which in turn will influence the objectivity of the readers' votes. Also, the higher ranked reviews would attract more eyeballs and therefore gain more people's votes. This mutual influence among labelers should be avoided when the votes are used as the evaluation standard.

3.1.3 Early Bird Bias

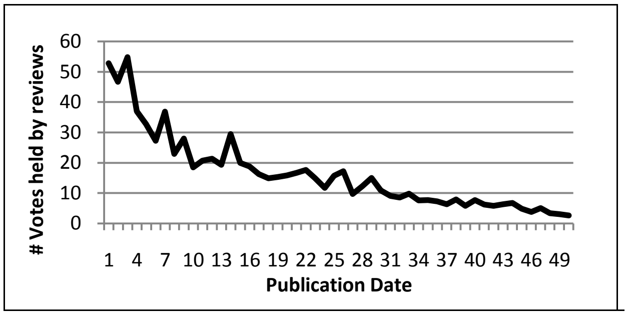


Figure 4. Dependency on publication date

Publication date can influence the accumulation of users' votes. In Figure 4, the *n*'*th* publication date represents the *n*'*th* month after the product is released. The number in the figure is averaged over all the digital cameras in the data set. We can observe a clear trend that the earlier a review is posted, the more votes it will get. This is simply because reviews posted earlier are exposed to users for a longer time. Therefore, some high quality reviews may get fewer users' vote because of later publication. We call this "early bird" bias.

3.2 Specification of Quality

Besides these aforementioned biases, using the raw rating from readers directly also fails to provide a clear guideline for what a good review consists of. In this section, we provide such a guideline, which we name as the specification (SPEC).

In the SPEC, we define four categories of review quality which represent different values of the reviews to users' purchase decision: "best review", "good review", "fair review", and "bad review". A generic description of the SPEC is as follows:

A *best* review must be a rather complete and detailed comment on a product. It presents several aspects of a product and provides convincing opinions with enough evidence. Usually a best review could be taken as the main reference that users only need to read before making their purchase decision on a certain product. The first review in Figure 5 is a *best* review. It presents several product features and provides convincing opinions with sufficient evidence. It is also in a good format for readers to easily understand. Note that we omit some words in the example to save the space.

A *good* review is a relatively complete comment on a product, but not with as much supporting evidence as necessary. It could be used as a strong and influential reference, but not as the only recommendation. The second review in Figure 5 is such an example.

A *fair* review contains a very brief description on a product. It does not supply detailed evaluation on the product, but only comments on some aspects of the product. For example, the third review in Figure 5 mainly talks about "*the delay between pictures*", but less about other aspects of the camera.

A *bad* review is usually an incorrect description of a product with misleading information. It talks little about a specific product but much about some general topics (e.g. photography). For example, the last review in Figure 5 talks about the topic of "*generic battery*", but does not specify any digital camera. A bad review is an "unhelpful" review that can be ignored.

Best Review:

I purchased this camera about six months ago after my Kodak Easyshare camera completely died on me. I did a little research and read only good things about this Canon camera so I decided to go with it because it was very reasonably priced (about $200). Not only did the camera live up to my expectations, it surpassed them by leaps and bounds! Here are the things I have loved about this camera:

BATTERY - this camera has the best battery of any digital camera I have ever owned or used. …

EASY TO USE - I was able to …

PICTURE QUALITY - all of the pictures I've taken and printed out have been great. …

FEATURES - I love the ability to quickly and easily …

LCD SCREEN - I was hoping …

SD MEMORY CARD - I was also looking for a camera that used SD memory cards. Mostly because…

I cannot stress how highly I recommend this camera. I will never buy another digital camera besides Canon again. And the A610 (as well as the A620 - the 7.0MP version) is the best digital camera I've ever used.

Good Review:

The Sony DSC "P10" Digital Camera is the top pick for CSC. Running against cameras like Olympus stylus, Canon Powereshot, Sony V1, Nikon, Fuji, and More. The new release of 5.0 mega pixels has shot prices for digital cameras up to $1000+. This camera I purchased through a Private Dealer cost me $400.86. The Retail Price is Running $499.00 to $599.00. Purchase this camera from a wholesale dealer for the best price $377.00. Great Photo Even in dim light w/o a flash. The p10 is very compact. Can easily fit into any pocket. The camera can record 90 minutes of mpeg like a home movie. There are a lot of great digital cameras on the market that shoot good pictures and video. What makes the p10 the top pick is it comes with a rechargeable lithium battery. Many use AA batteries, the digital camera consumes theses AA batteries in about two hours time while the unit is on. That can add continuous expense to the camera. It's also the best resolution on the market. 6.0 megapix is out, though only a few. And the smallest that we found. Also the best price for a major brand.

Fair Review:

There is nothing wrong with the 2100 except for the very noticeable delay between pics. The camera's digital processor takes about 5 seconds after a photo is snapped to ready itself for the next one. Otherwise, the optics, the 3X optical zoom and the 2 megapixel resolution are fine for anything from Internet apps to 8" x 10" print enlarging. It is competent, not spectacular, but it gets the job done at an agreeable price point.

Bad Review:

I want to point out that you should never buy a generic battery, like the person from San Diego who reviewed the S410 on May 15, 2004, was recommending. Yes you'd save money, but there have been many reports of generic batteries exploding when charged for too long. And don't think if your generic battery explodes you can sue somebody and win millions. These batteries are made in sweatshops in China, India and Korea, and I doubt you can find anybody to sue. So play it safe, both for your own sake and the camera's sake. If you want a spare, get a real Canon one.

Figure 5. Example reviews

3.3 Annotation of Quality

According to the SPEC defined above, we built a ground-truth from the Amazon data set. We randomly selected 100 digital cameras and 50 reviews for each camera. Totally we have 4,909 reviews since some digital cameras have fewer than 50 unique reviews. Then we hired two annotators to label the reviews with the SPEC as their guideline. As the result, we have two independent copies of annotations on 4,909 reviews, with the labels of "best", "good", "fair", and "bad". Table 1 shows the confusion matrix between the two copies of annotation. The value of the *kappa* statistic (Cohen, 1960) calculated from the matrix is 0.8142. This shows that the two annotators achieved highly consistent results by following the SPEC, although they worked independently.

Annotation 1	Annotation 2				
	best	*good*	*fair*	*bad*	total
best	294	44	2	0	340
good	66	639	113	0	818
fair	0	200	1,472	113	1,785
bad	1	2	78	1,885	1,966
total	361	885	1,665	1,998	4,909

Table 1. Confusion matrix bet. the annotations

In order to examine the difference between our annotations and Amazon ground-truth, we evaluate the Amazon ground-truth against the annotations, with the measure of "error rate of preference pairs" (Herbrich et al, 1999).

$$ErrorRate = \frac{|incorrect\ preference\ pairs\ |}{|all\ preference\ pairs|} \quad (1)$$

where the "*preference pair*" is defined as a pair of reviews with a order. For example, a *best* review and a *good* review correspond to a preference pair with the order of "*best* review preferring to *good* review". The "*all preference pairs*" are collected from one of the annotations (the annotation 1 or the annotation 2) by ignoring the pairs from the same category. The "*incorrect preference pairs*" are the *preference pair*s collected from the Amazon ground-truth but not with the same order as that in the *all preference pairs*. The order of the *preference pair* collected from the Amazon ground-truth is evaluated on the basis of the *percentage* score as described in Section 3.1.

The error rate of preference pairs based on the annotation 1 and that based on the annotation 2 are 0.448 and 0.446, respectively, averaged over 100 digital cameras. The high error rate of preference pairs demonstrates that the Amazon ground-truth diverges from the annotations (our ground-truth) significantly.

To discover which kind of ground-truth is more reasonable, we ask an additional annotator (the third annotator) to compare these two kinds of ground-truth. More specifically, we randomly selected 100 preference pairs whose orders the two kinds of ground-truth don't agree on (called incorrect preference pairs in the evaluation above). As for our ground-truth, we choose the Annotation 1 in the new test. Then, the third annotator is asked to assign a preference order for each selected pair. Note that the third annotator is blind to both our specification and the existing preference order. Last, we evaluate the two kinds of ground-truth with the new annotation. Among 100 pairs, our ground-truth agrees to the new annotation on 85 pairs while the Amazon ground-truth agrees to the new annotation on 15 pairs. To confirm the result, yet another annotator (the fourth annotator) is called to repeat the same annotation independently as the third one. And we obtain the same statistical result (85 vs. 15) although the fourth annotator does not agree with the third annotator on some pairs.

In practice, we treat the reviews in the first three categories ("*best*", "*good*" and "*fair*") as high-quality reviews and those in the "*bad*" category as low-quality reviews, since our goal is to identify low quality reviews that should not be considered when creating product review summaries.

4 Classification of Product Reviews

We employ a statistical machine learning approach to address the problem of detecting low-quality products reviews.

Given a training data set $D = \{x_i, y_i\}_1^n$, we construct a model that can minimize the error in prediction of y given x (generalization error). Here $x_i \in X$ and $y_i = \{high\ quality\ , low\ quality\}$ represents a product review and a label, respectively. When applied to a new instance *x*, the model predicts the corresponding *y* and outputs the score of the prediction.

4.1 The Learning Model

In our study, we focus on differentiating low-quality product reviews from high-quality ones. Thus, we treat the task as a binary classification problem.

We employ SVM (Support Vector Machines) (Vapnik, 1995) as the model of classification. Given an instance *x* (product review), SVM assigns a score to it based on

$$f(x) = w^T x + b \quad (2)$$

where *w* denotes a vector of weights and *b* denotes an intercept. The higher the value of *f(x)* is, the higher the quality of the instance *x* is. In classification, the sign of *f(x)* is used. If it is positive, then *x* is classified into the positive category (high-quality reviews), otherwise into the negative category (low-quality reviews).

The construction of SVM needs labeled training data (in our case, the categories are "high-quality reviews" and "low-quality reviews"). Briefly, the learning algorithm creates the "hyper plane" in (2), such that the hyper plane separates the positive and negative instances in the training data with the largest "margin".

4.2 Product Feature Resolution

Product features (e.g., "image quality" for digital camera) in a review are good indicators of review quality. However, different product features may refer to the same meaning (e.g., "*battery life*" and "*power*"), which will bring redundancy in the study. In this paper, we formulize the problem as the "resolution of product features". Thus, the problem is reduced to how to determine the equivalence of a product feature in different forms.

In (Hu and Liu, 2004), the matching of different product features is mentioned briefly and addressed by fuzzy matching. However, there exist many cases where the method fails to match the multiple mentions, e.g., "*battery life*" and "*power*", because it only considers string similarity. In this paper we propose to resolve the problem by leveraging two kinds of evidence: one is "surface string" evidence, the other is "contextual evidence".

We use *edit distance* (Ukkonen, 1985) to compare the similarity between the surface strings of two mentions, and use *contextual similarity* to reflect the semantic similarity between two mentions.

When using contextual similarity, we split all the reviews into sentences. For each mention of a product feature, we take it as a query and search for all the relevant sentences. Then we construct a vector for the mention, by taking each unique term in the relevant sentences as a dimension of the vector. The cosine similarity between two vectors of mentions is then present to measure the contextual similarity between two mentions.

4.3 Feature Development for Learning

To detect low-quality reviews, our proposed approach explores three aspects of product reviews, namely informativeness, subjectiveness, and readability. We denote the features employed for learning as "learning features", discriminative from the "product features" we discussed above.

4.3.1 Features on Informativeness

As for informativeness, the resolution of product features is employed when we generate the learning features as listed below. Pairs mapping to the same product feature will be treated as the same product feature, when we calculate the frequency and the number of product features. We apply the approach proposed in (Hu and Liu, 2004) to extract product features.

We also use a list of product names and a list of brand names to generate the learning features. Both lists can be collected from the Amazon site because they are relatively stable within a time interval.

The learning features on the informativeness of a review are as follows.

➢ **Sentence level (SL)**

- The number of sentences in the review
- The average length of sentences
- The number of sentences with product features

➢ **Word level (WL)**

- The number of words in the review
- The number of products (e.g., DMC-FZ50, EX-Z1000) in the review
- The number of products in the title of a review
- The number of brand names (e.g., Canon, Sony) in the review
- The number of brand names in the title of a review

➢ **Product feature level (PFL)**

- The number of product features in the review
- The total frequency of product features in the review
- The average frequency of product features in the review
- The number of product features in the title of a review
- The total frequency of product features in the title of a review

4.3.2 Features on Readability

We make use of several features at paragraph level which indicate the underlying structure of the reviews. These features include,

- The number of paragraphs in the review
- The average length of paragraphs in the review
- The number of paragraph separators in the review

Here, we refer to the keywords, such as "Pros" vs. "Cons" as "paragraph separators". The keywords usually appear at the beginning of paragraphs for categorizing two contrasting aspects of a product. We extract the nouns and noun phrases at the beginning of each paragraph from the 4,909 reviews and use the most frequent 30 pairs of keywords as paragraph separators. Table 2 provides some examples of the extracted separators.

Separators		***Separators***	
Positive	***Negative***	***Positive***	***Negative***
Pros	Cons	The Good	The Bad
Strength	Weakness	Thumb up	Bummer
PLUSES	MINUSES	Positive	Negative
Advantages	Drawbacks	Likes	Dislikes
The upsides	Downsides	GOOD THINGS	BAD THINGS

Table 2. Examples of paragraph separators

4.3.3 Features on Subjectiveness

We also take the subjectiveness of reviews into consideration. Unlike previous work (Kim et al, 2006; Zhang and Varadarajan, 2006) using shallow syntactic information directly, we use a sentiment analysis tool (Hu and Liu, 2004) which aggregates a set of shallow syntactic information. The tool is a classifier capable of determining the sentiment polarity of each sentence. We create three learning features regarding the subjectiveness of reviews.

- The percentage of positive sentences in the review
- The percentage of negative sentences in the review
- The percentage of subjective sentences (regardless of positive or negative) in the review

5 Experiments

In this section, we describe our experiments with the proposed classification-based approach to low-quality review detection, and its effectiveness on the task of opinion summarization.

5.1 Detecting Low-quality Reviews

In our proposed approach, the problem of assessing quality of reviews is formalized as a binary classification problem. We conduct experiments by taking reviews in the categories of “best”, “good”, and “fair” as high-quality reviews and those in the “bad” category as low-quality reviews.

As for classification model, we utilize the SVMLight toolkit (Joachims, 2004). We randomly divide the 100 queries of digital cameras into two sets, namely a training set of 50 queries and a test set of 50 queries. For the two copies of annotations, we use the same division. We use the training set from “annotation 1” to train the model and apply the model to the test sets from both “annotation 1” and “annotation 2”, respectively. Table 3 reports the accuracies of our approach to review classification. The accuracy is defined as the percentage of correctly classified reviews.

We take the approach that utilizes only the category of features on sentence level (SL) as the baseline, and incrementally add other categories of features on informativeness, readability and subjectiveness. We can see that both the features on word level (WL) and those on product feature level (PFL) can improve the performance of classification much. The features on readability can still increase the accuracy although the contribution is much less. The features on subjectiveness, however, make no contribution.

Feature Category		Annotation1	Annotation2
Informative-ness	SL	73.59%	72.81%
	WL	80.41%	79.15%
	PFL	83.30%	82.37%
Readability		83.93%	82.91%
Subjectiveness		83.84%	82.96%

Table 3. Low-quality reviews detection

We also conduct a more detailed analysis on each individual feature. Two categories of features on “title” and “brand name” have poor performance, which is due to the lack of information in the title and the low coverage of brand names in a review, respectively.

5.2 Summarizing Sentiments of Reviews

One potential application of low-quality review detection is the opinion summarization of reviews.

The process of opinion summarization of reviews with regards to a query of a product consists of the following steps (Liu et al, 2005):

1. From each of the reviews, identify every text segment with opinion in the review, and determine the polarities of the opinion segments.
2. For each product feature, generate a positive opinion set and a negative opinion set of opinion segments, denoted as $\mathrm{POS}(f)$ and $\mathrm{NOS}(f)$.
3. For each product feature, aggregate the numbers of segments in $\mathrm{POS}(f)$ and$\mathrm{NOS}(f)$, as opinion summarization on the product feature.

In this process, all the reviews contribute the same. However, different reviews do hold different authorities. A positive/negative opinion from a high-quality review should not have the same weight as that from a low-quality review.

We use a two-stage approach to enhance the reliability of summarization. That is, we add a process of low-quality review detection before the summarization process, so that the summarization result is obtained based on the high-quality reviews only. We are to demonstrate how much difference the proposed two-stage approach can bring into the opinion summarization.

We use the best classification model trained as described in Section 5.1 to filter low-quality reviews, and do summarization on the high-quality reviews associated to the 50 test queries. We denote the proposed approach and the old approach as "two-stage" and "one-stage", respectively. Due to the limited space, we only give a visual comparison of the two approaches on "image quality" in Figure 6. The upper figure shows the summarization of positive opinions and the lower figure shows that of negative opinions. From the figures we can see that the two-stage approach preserves fewer text segments as the result of filtering out many low-quality product reviews.

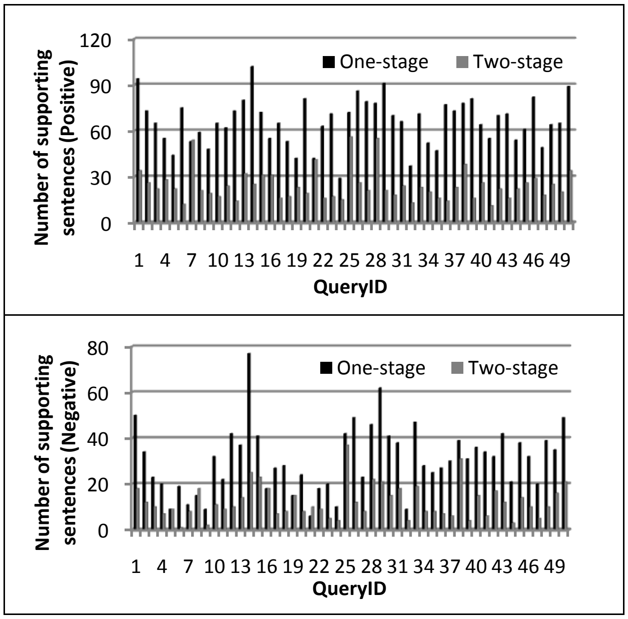


Figure 6. Summarization on "image quality"

To show the comparison on more features in a compressed space, we give the statistic ratio of change between two approaches instead. As for the evaluation measure, we define "*RatioOfChange*" (*ROC)* on a feature *f* as,

$$\mathrm{ROC}(f) = \frac{\mathrm{Rate}_{\mathrm{one-stage}}\,(f) - \mathrm{Rate}_{\mathrm{two-stage}}\,(f)}{\mathrm{Rate}_{\mathrm{one-stage}}\,(f)} \qquad (3)$$

where Rate $_{*}(f)$ is defined as,

$$\mathrm{Rate}_{*}(f) = \frac{|\mathrm{POS}(f)|}{|\mathrm{POS}(f)| + |\mathrm{NOS}(f)|} \qquad (4)$$

Table 4 shows some statistic results on *ROC* on five product features, namely "image quality"(IQ), "battery", "LCD screen" (LCD), "flash" and "movie mode" (MM). The values in the cells are the percentage of queries whose *ROC* is larger/smaller than the respective thresholds. We can see that a large portion of queries have big changes on the values of *ROC*. This means that the result achieved by the two-stage approach is substantially different from that achieved by the one-stage approach.

%Query	***RatioOfChange* (+)**					
	>0.30	**>0.25**	**>0.20**	**>0.15**	**>0.10**	**>0.05**
IQ	2%	4%	4%	10%	14%	22%
Battery	10%	14%	18%	30%	38%	50%
LCD	12%	18%	20%	22%	24%	28%
Flash	6%	10%	16%	20%	26%	42%
MM	6%	8%	8%	12%	18%	26%
%Query	***RatioOfChange* (-)**					
	<-0.30	**<-0.25**	**<-0.20**	**<-0.15**	**<-0.10**	**<-0.05**
IQ	4%	6%	10%	14%	18%	44%
Battery	2%	4%	4%	10%	14%	22%
LCD	4%	4%	8%	12%	22%	28%
Flash	4%	6%	8%	16%	18%	28%
MM	8%	10%	16%	18%	34%	42%

Table 4. *RatioOfChange* on five features

There is no standard way to evaluate the quality of opinion summarization as it is rather a subjective problem. In order to demonstrate the impact of the two-stage approach, we turn to external authoritative sources other than Amazon.com as the objective evaluation reference. We observe that CNET[2] provides a professional "*editor's review*" for many products, which gives a rating in the range of 1~10 on product features. 9 digital cameras out of the 50 test queries are found to have the editor's rating on "image quality" at CNET. We use this rating to compare with the results of our opinion summarization. We rescale the *Rate* scores obtained by both the one-stage approach and the two-stage approach into the range of 1-10 in order to perform the comparison.

Figure 7 provides the visual comparison. We can see that the result achieved by the two-stage approach has a much better (closer) resemblance to CNET rating than one-stage approach does. This indicates that our two-stage approach can achieve a more consistent summarization result to the professional evaluations by the editors. Although the CNET rating is not the absolute standard for product evaluation, it provides a professional yet objective evaluation of the products. Therefore, the experimental results demonstrate that our proposed approach could achieve more reliable opinion summarization which is closer to the generic evaluation from authoritative sources.

[2] http://www.cnet.com

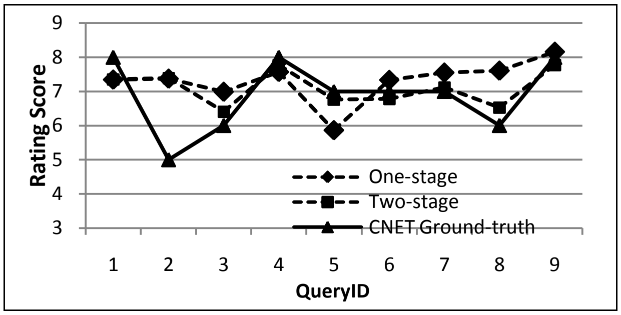


Figure 7. Comparison with CNET rating

6 Conclusion

In this paper, we studied the problem of detecting low-quality product reviews. Our contribution can be summarized in two-fold: (1) we discovered **three types of biases** in the ground-truth used extensively in the existing work, and proposed a specification on the quality of product reviews. The three biases that we discovered are *imbalance vote bias*, *winner circle bias*, and *early bird bias*. (2) Rooting on the new ground-truth (conforming to the proposed specification), we proposed a classification-based approach to low-quality product review detection, which yields better performance of **opinion summarization**.

We hope to explore our future work in several areas, such as further consolidating the new ground-truth from different points of view and verifying the effectiveness of low-quality review detection with other applications.

References

Jacob Cohen. 1960. A coefficient of agreement for nominal scales, *Educational and Psychological Measurement* 20: 37–46.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. *WWW'03*.

Harris Drucker, Chris J.C., Burges Linda Kaufman, Alex Smola and Vladimir Vapnik. 1997. Support vector regression machines. *Advances in Neural Information Processing Systems.*

Christiane Fellbaum. 1998. WordNet: an Electronic Lexical Database, MIT Press.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support Vector Learning for Ordinal Regression. *In Proc. of the 9th International Conference on Artificial Neural Networks*.

Minqing Hu and Bing Liu. 2004a. Mining and Summarizing Customer Reviews. *KDD'04*.

Minqing Hu and Bing Liu. 2004b. Mining Opinion Features in Customer Reviews. *AAAI'04*.

Kalervo Jarvelin & Jaana Kekalainen. 2000. IR: evaluation methods for retrieving highly relevant documents. *SIGIR'00*.

Nitin Jindal and Bing Liu. 2006. Identifying Comparative Sentences in Text Documents. *SIGIR'06*.

Nitin Jindal and Bing Liu. 2006. Mining comparative sentences and relations. *AAAI'06*.

Thorsten Joachims. SVMlight -- Support Vector Machine. http://svmlight.joachims.org/, 2004.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, Marco Pennacchiotti. 2006. Automatically Assessing Review Helpfulness. *EMNLP'06*.

Dekang Lin. 1998, Automatic retrieval and clustering of similar words. *COLING-ACL'98.*

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. *WWW '0*5.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *ACL'04*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *ACL'0*5.

Bo Pang and Lillian Lee, and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *EMNLP'02*.

Ana-Maria Popescu and O Etzioni. 2005. Extracting product features and opinions from reviews. *HLT-EMNLP'05.*

Peter D. Turney. 2001. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *ACL'02*

Esko Ukkonen. 1985. Algorithms for approximate string matching. *Information and Control*, pp. 100 – 118.

Vladimir N. Vapnik. 1995. The Nature of Statistical Learning Theory. Springer.

Zhu Zhang and Balaji Varadarajan. 2006. Utility Scoring of Product Reviews. CIKM'06

Improving Statistical Machine Translation Performance by Training Data Selection and Optimization

Yajuan Lü, Jin Huang and Qun Liu
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100080, China
{lvyajuan, huangjin,liuqun}@ict.ac.cn

Abstract

Parallel corpus is an indispensable resource for translation model training in statistical machine translation (SMT). Instead of collecting more and more parallel training corpora, this paper aims to improve SMT performance by exploiting full potential of the existing parallel corpora. Two kinds of methods are proposed: offline data optimization and online model optimization. The offline method adapts the training data by redistributing the weight of each training sentence pairs. The online method adapts the translation model by redistributing the weight of each predefined submodels. Information retrieval model is used for the weighting scheme in both methods. Experimental results show that without using any additional resource, both methods can improve SMT performance significantly.

1 Introduction

Statistical machine translation relies heavily on the available training data. Typically, the more data is used to estimate the parameters of the translation model, the better it can approximate the true translation probabilities, which will obviously lead to a higher translation performance. However, large corpora are not easily available. The collected corpora are usually from very different areas. For example, the parallel corpora provided by LDC come from quite different domains, such as Hongkong laws, Hangkong Hansards and Hongkong news. This results in the problem that a translation system trained on data from a particular domain(e.g. Hongkong Hansards) will perform poorly when translating text from a different domain(e.g. news articles). Our experiments also show that simply putting all these domain specific corpora together will not always improve translation quality. From another aspect, larger amount of training data also requires larger computational resources. With the increasing of training data, the improvement of translation quality will become smaller and smaller. Therefore, while keeping collecting more and more parallel corpora, it is also important to seek effective ways of making better use of available parallel training data.

There are two cases when we train a SMT system. In one case, we know the target test set or target test domain, for example, when building a specific domain SMT system or when participating the NIST MT evaluation[1]. In the other case, we are unaware of any information of the testing data. This paper presents two methods to exploit full potential of the available parallel corpora in the two cases. For the first case, we try to optimize the training data offline to make it match the test data better in domain, topic and style, thus improving the translation performance. For the second case, we first divide the training data into several domains and train submodels for each domain. Then, in the translation process, we try to optimize the predefined models according to the online input source sentence. Information retrieval model is used for similar sentences retrieval in both methods. Our preliminary experiments show that both methods can improve SMT performance without using any additional data.

[1] http://www.nist.gov/speech/tests/mt/

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 343–350, Prague, June 2007. ©2007 Association for Computational Linguistics

The remainder of this paper is organized as follows: Section 2 describes the offline data selection and optimization method. Section 3 describes the online model optimization method. The evaluation and discussion are given in section 4. Related work is introduced before concluding.

2 Offline training data optimization

In offline training data optimization, we assume that the target test data or target test domain is known before building the translation model. We first select sentences similar to the test text using information retrieval method to construct a small and adapted training data. Then the extracted similar subset is used to optimize the distribution of the whole training data. The adapted and the optimized training data will be used to train new translation models.

2.1 Similar data selection using TF-IDF

We use information retrieval method for similar data retrieval. The standard TF-IDF (Term Frequency and Inverse Document Frequency) term weighting scheme is used to measure the similarity between the test sentence and the training sentence.

TF-IDF is a similarity measure widely used in information retrieval. Each document D_i is represented as a vector $(w_{i1}, w_{i2}, ..., w_{in})$, n is the size of the vocabulary. w_{ij} is calculate as follows:

$$w_{ij} = tf_{ij} \times \log(idf_j)$$

where,

tf_{ij} is the term frequency(TF) of the *j*-th word in the vocabulary in the document D_i, i.e. the number of occurrences;

idf_j is the inverse document frequency(IDF) of the *j*-th word calculated as below:

$$idf_j = \frac{\#documents}{\#documents\ containing\ \text{j - th term}}.$$

The similarity between two documents is then defined as the cosine of the angle between the two vectors.

We perform information retrieval using the Lemur toolkit[2]. The source language part of the parallel training data is used as the document collection. Each sentence represents one document. Each sentence from the test data or test domain is used as one separate query. In the sentence retrieval process, both the query and the document are converted into vectors by assigning a term weight to each word. Then the cosine similarity is calculated proportional to the inner product of the two vectors. All retrieved sentences are ranked according to their similarity with the query. We pair each of the retrieved sentences with the corresponding target part and the top N most similar sentences pairs are put together to form an adapted parallel data. N ranges from one to several thousand in our experiments. Since Lemur toolkit gives the similarity score for each retrieved sentences, it is also possible to select the most similar sentences according to the similarity score.

Note that the selected similar data can contain duplicate sentences as the top N retrieval results for different test sentences can contain the same training sentences. The duplicate sentences will force the translation probability towards the more often seen words. Intuitively, this could help. In experiment section, we will compare experimental results by keeping or removing duplicates to see how the duplicate sentences affect the translations.

The selected subset contains the similar sentences with the test data or test domain. It matches the test data better in domain, topic and style. Hopefully, training translation model using this adapted parallel data may helpful for improving translation performance. In addition, the translation model trained using the selected subset is usually much smaller than that trained using the whole translation data. Limiting the size of translation model is very important for some real applications. Since SMT systems usually require large computation resource. The complexity of standard training and decoding algorithm depends mainly on the size of the parallel training data and the size of the translation model. Limiting the size of the training data with the similar translation performance would also reduce the memories and speed up the translations.

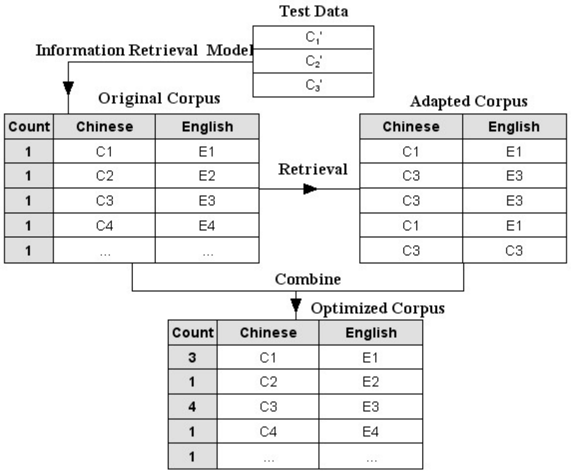
In the information retrieval process, we only use the source language part for document indexing and query generating. It is easy to get source part of the test data. This is different from the common language model adaptation methods, which have to do at lease one pass machine translation to get the candidate English translation as query(Zhao 2004, Zhang 2006). So our method has the advantage that it is independent from the quality of baseline translation system.

[2] http://www.cs.cmu.edu/~lemur/

2.2 Training data optimization

There are two factors on training data that influence the translation performance of SMT system: the scale and the quality. In some sense, we improve the quality of the training data by selecting the similar sentence to form an adapted training set. However, we also reduce the scale of the training data at the same time. Although this is helpful for some small device applications, it is also possible to induce the data sparseness problem. Here, we introduce a method to optimize between the scale and the quality of the training data.

The basic idea is that we still use all the available training data; by redistributing the weight of each sentence pairs we adapt the whole training data to the test domain. In our experiments, we simply combine the selected small similar subset and the whole training data. The weights of each sentence pairs are changed accordingly. Figure 1 shows the procedure of the optimization.



Count	Chinese	English
1	C1	E1
1	C2	E2
1	C3	E3
1	C4	E4
1	...	...

Chinese	English
C1	E1
C3	E3
C3	E3
C1	E1
C3	C3

Count	Chinese	English
3	C1	E1
1	C2	E2
4	C3	E3
1	C4	E4
1	...	...

Figure 1. Training data optimization

As can be seen, through the optimization, the weight of the similar sentence pairs are increased, while the general sentence pairs still have an ordinary weight. This make the translation model inclined to give higher probabilities to the adapted words, and at the same time avoid the data sparseness problem. Since we only change the weight of the sentence pairs, and no new training data is introduced, the translation model size trained on the optimized data will keep as the same as the original one. We use GIZA++ toolkit[3] for word alignment training in the training process. The input training file formats for GIZA++ is as follows: Each training sentence pair is stored in three lines. The first line is the number of times this sentence pair occurred. The second line is the source sentence where each token is replaced by its unique integer id and the third is the target sentence in the same format. To deal with our optimized training data, we only need to change the number of sentence pairs in the first line accordingly. This will not call for extra training time and memory for the whole training process.

It might be beneficial to investigate other sophisticated weighting schemes under the similar idea, such as to give more precise fractional weights to the sentences according the retrieval similarity scores.

3 Online model optimization

In most circumstances, we don't know exactly the test data or the test domain when we train a machine translation system. This results in the fact that the performance of the translation system highly depends on the training data and the test data it is used in. To alleviate this blindfold status and maximize the potential of the available training corpora, we propose a novel online model optimization method.

The basic idea is that: several candidate translation models are prepared in training stage. In particularly, a general model is also prepared. Then, in the translation process, the similarity between the input sentence and the predefined models is calculated online to get the weights of each model. The optimized model is used to translate the input sentence.

There are two problems in the method: how to prepare submodels in training process and how to optimize the model weight online in translation process.

3.1 Prepare the submodels

There are several ways to prepare submodels in training process. If the training data comes from very different sources, we can divide the data according to its origins. Otherwise, we can use clustering method to separate the training corpus into several classes. In addition, our offline data adaptation method can also be used for submodel preparation. For each candidate domain, we can use the

[3] http://www.fjoch.com/GIZA++.html

source side of a small corpus as queries to extract a domain specific training set. In this case, a sentence pair in the training data may occur in several sub training data, but this doesn't matter. The general model is used when the online input is not similar to any prepared submodels. We can use all available training data to train the general model since generally larger data can get better model even there are some noises.

3.2 Online model weighting

We also use TF-IDF information retrieval method for online model weighting. The procedure is as follows:

For each input sentence:

1. *Do IR on training data collection, using the input sentence as query.*
2. *Determine the weights of submodels according to the retrieved sentences.*
3. *Use the optimized model to translate the sentence.*

The information retrieval process is the same as the offline data selection except that each retrieved sentence is attached with the sub-corpus information, i.e. it belongs to which sub-models in the training process.

With the sub-corpus information, we can calculate the weights of submodels. We get the top N most similar sentences, and then calculate proportions of each submodel's sentences. The proportion can be calculated use the count of the sentences or the similarity score of the sentences. The weight of each submodel can be determined according to the proportions.

Our optimized model is the log linear interpolation of the sub-models as follows:

$$\hat{p}(e \mid c) = p_0(e \mid c)^{\delta_0} \times \prod_{i=1}^{M} p_i(e \mid c)^{\delta_i}$$

$$\hat{e} = \arg\max_{e}(\delta_0 \log(p_0(e \mid c)) + \sum_{i=1}^{M} \delta_i \log(p_i(e \mid c)))$$

where, p_0 is the probability of general model, p_i is the probability of submodel *i*. δ_0 is the weight of general model. δ_i is the weight of submodel *i*. Each model *i* is also implemented using log linear model in our SMT system. So after the log operation, the submodels are interpolated linearly.

In our experiments, the interpolation factor δ_i is determined using the following four simple weighting schemes:

Weighting scheme 1:

$$\delta_0 = 0; \quad \delta_{max\_model} = 1; \quad \delta_{i \neq max\_model} = 0;$$

Weighting scheme 2:

```
if Proportion(max_model) > 0.5
    Use weighting scheme1;
else
    $\delta_0 = 1$;  $\delta_i = 0$;
```

Weighting scheme 3:

$$\delta_0 = 0;$$

$$\delta_i = \text{Proportion } (model_i);$$

Weighting scheme 4:

```
if Proportion(max_model) > 0.5
    Use weighting scheme3;
else
    $\delta_0 = 0.5$;
    $\delta_i = 0.5 \times$ Proportion $(model_i)$;
```

where, *model*i is the *i*-th submodel, $i = (1...M)$. Proportion (*model*$_i$) is the proportion of *model*$_i$ in the retrieved results. We use count for proportion calculation. *max_model* is the submodel with the max proportion score.

The training and translation procedure of online model optimization is illustrated in Figure 2.

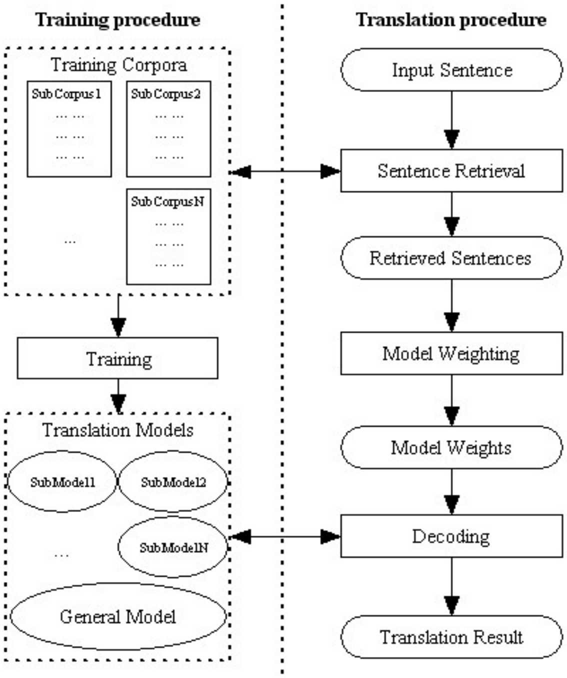


Figure 2. Online model optimization

The online model optimization method makes it possible to select suitable models for each individual test sentence. Since the IR process is done on a fixed training data, the size of the index data is quite small compared with the web IR. The IR process will not take much time in the translation.

4 Experiments and evaluation

4.1 Experimental setting

We conduct our experiments on Chinese-to-English translation tasks. The baseline system is a variant of the phrase-base SMT system, implemented using log-linear translation model (He et al. 2006). The baseline SMT system is used in all experiments. The only difference between them is that they are trained on different parallel training data.

In training process, we use GIZA++[4] toolkit for word alignment in both translation directions, and apply "grow-diag-final" method to refine it (Koehn et al., 2003). We change the preprocess part of GIZA++ toolkit to make it accept the weighted training data. Then we use the same criterion as suggested in (Zens et al., 2002) to do phrase extraction. For the log-linear model training, we take minimum-error-rate training method as described in (Och, 2003). The language model is trained using Xinhua portion of Gigaword with about 190M words. SRI Language Modeling toolkit[5] is used to train a 4-gram model with modified Kneser-Ney smoothing(Chen and Goodman, 1998). All experiments use the same language model. This ensures that any differences in performance are caused only by differences in the parallel training data.

Our training data are from three LDC corpora as shown in Table 1. We random select 200,000 sentence pairs from each corpus and combine them together as the baseline corpus, which includes 16M Chinese words and 19M English words in total. This is the usual case when we train a SMT system, i.e. we simply combine all corpora from different origins to get a larger training corpus.

We use the 2002 NIST MT evaluation test data as our development set, and the 2005 NIST MT test data as the test set in offline data optimization experiments. In both data, each sentence has four human translations as references. The translation quality is evaluated by BLEU metric (Papineni et al., 2002), as calculated by mteval-v11b.pl[6] with case-sensitive matching of n-grams.

Corpus	LDC No.	Description	# sent. pairs
FBIS	LDC2003E14	FBIS Multilanguage Texts	200000
HK_Hansards	LDC2004T08	Hong Kong Hansards Text	200000
HK_News	LDC2004T08	Hong Kong News Text	200000
Baseline	-	All above data	600000

Table 1. Training corpora

4.2 Baseline experiments

We first train translation models on each sub training corpus and the baseline corpus. The development set is used to tune the feature weights. The results on test set are shown in Table 2.

System	BLEU on dev set	BLEU on test set
FBIS	0.2614	0.2331
HK_Hansards	0.1679	0.1624
HK_News	0.1748	0.1608
Baseline	0.2565	0.2363

Table 2. Baseline results

From the results we can see that although the size of each sub training corpus is similar, the translation results from the corresponding system are quite different on the same test set. It seems that the FBIS corpus is much similar to the test set than the other two corpora. In fact, it is the case. The FBIS contains text mainly from China mainland news stories, while the 2005 NIST test set also include lots of China news text. The results illustrate the importance of selecting suitable training data.

When combining all the sub corpora together, the baseline system gets a little better result than the sub systems. This indicates that larger data is useful even it includes some noise data. However, compared with the FBIS corpus, the baseline corpus contains three times larger data, while the improvement of translation result is not significant. This indicates that simply putting different corpora together is not a good way to make use of the available corpora.

[4] http://www.fjoch.com/GIZA++.html

[5] http://www.speech.sri.com/projects/srilm/

[6] http://www.nist.gov/speech/tests/mt/resources/scoring.htm

4.3 Offline data optimization experiments

We use baseline corpus as initial training corpus, and take Lemur toolkit to build document index on Chinese part of the corpus. The Chinese sentences in development set and test set are used as queries. For each query, N = 100, 200, 500, 1000, 2000 similar sentences are retrieved from the indexed collection. The extracted similar sentence pairs are used to train the new adapted translation models. Table 3 illustrates the results. We give the distinct pair numbers for each adapted set and compare the size of the translation models. To illustrate the effect of duplicate sentences, we also give the results with duplicates and without duplicates (distinct).

System	Distinct pairs	Size of trans model	BLEU on duplicates	BLEU on distinct
Baseline	600000	2.41G	0.2363	0.2363
Top100	91804	0.43G	0.2306	0.2346
Top200	150619	0.73G	0.2360	0.2345
Top500	261003	1.28G	0.2415	0.2370
Top1000	357337	1.74G	0.2463	0.2376
Top2000	445890	2.11G	0.2351	0.2346

Table 3. Offline data adaptation results

The results show that:

1. By using similar data selection, it is possible to use much smaller training data to get comparable or even better results than the baseline system. When N=200, using only 1/4 of the training data and 1/3 of the model size, the adapted translation model achieves comparable result with the baseline model. When N=500, the adapted model outperforms the baseline model with much less training data. The results indicate that relevant data is better data. The method is particular useful for SMT applications on small device.

2. In general, using duplicate data achieves better results than using distinct data. This justifies our idea that give a higher weight to more similar data will benefit.

3. With the increase of training data size, the translation performance tends to improve also. However, when the size of corpus achieves a certain scale, the performance may drop. This maybe because that with the increase of the data, noisy data may also be included. More and more included noises may destroy the data. It is necessary to use a development set to determine an optimal size of N.

We combine each adapted data with the baseline corpus to get the optimized models. The results are shown in Table 4. We also compare the adapted models (TopN) and the optimized models (TopN+) in the table.

Without using any additional data, the optimized models achieve significant better results than the baseline model by redistributing the weight of training sentences. The optimized models also outperform adapted models when the size of the adapted data is small since they make use of all the available data which decrease the influence of data sparseness. However, with the increase of the adapted data, the performance of optimized models is similar to that of the adapted models.

System	Distinct pairs	BLEU on TopN	BLEU on TopN+
Baseline	600000	0.2363	0.2363
Top100+	600000	0.2306	0.2387
Top200+	600000	0.2360	0.2443
Top500+	600000	0.2415	0.2461
Top1000+	600000	0.2463	0.2431
Top2000+	600000	0.2351	0.2355

Table 4. Offline data optimization results

4.4 Online model optimization experiments

Since 2005 NIST MT test data tends bias to FBIS corpus too much, we build a new test set to evaluate the online model optimization method. We randomly select 500 sentences from extra part of FBIS, HK_Hansards and HK_News corpus respectively (i.e the selected 1500 test sentences are not included in any of the training set). The corresponding English part is used as translation reference. Note that there is only one reference for each test sentence. We also include top 500 sentence and their first reference translation of 2005 NIST MT test data in the new test set. So in total, the new test contains 2000 test sentences with one translation reference for each sentence. The test set is used to simulate SMT system's online inputs which may come from various domains.

The baseline translation results are shown in Table 5. We also give results on each sub test set (denotes as Xcorpus_part). Please note that the absolute BLEU scores are not comparable to the previous experiments since there is only one reference in this test set.

As expected, using the same domain data for training and testing achieves the best results as indicate by **bold fonts**. The results demonstrate again that relevant data is better data.

To test our online model optimization method, we divide the baseline corpus according to the origins of sub corpus. That is, the FBIS, HK_ Hansards and HK_News models are used as three sub-models and the baseline model is used as general model. The four weighting schemes described in section 3.2 are used as online weighting schemes individually. The experimental results are shown in Table 6. S_*i* indicates the system using weighting scheme *i*.

System **Test data**	**FBIS**	**HK_ Hansards**	**HK_ News**	**Baseline**
FBIS-part	**0.1096**	0.0687	0.0622	0.1030
HK_Hans_part	0.0726	**0.0918**	0.0846	0.0897
HK_News_part	0.0664	0.0801	**0.0936**	0.0870
MT05_part	0.1130	0.0805	0.0776	0.1116
Whole test set	0.0937	0.0799	0.0781	0.0993

Table 5. Baseline results on new test set

System **Test data**	**S_1**	**S_2**	**S_3**	**S_4**
FBIS-part	0.1090	0.1090	0.1089	0.1089
HK_Hans_part	0.0906	0.0903	0.0902	0.0902
HK_News_part	0.0952	0.0950	0.0933	0.0934
MT05_part	0.1119	0.1123	0.1149	0.1151
Whole test set	0.1034	0.1034	0.1038	0.1038

Table 6. Online model optimization results

Different weighting schemes don't show significant improvements from each other. However, all the four weighting schemes achieve better results than the baseline system. The improvements are shown not only on the whole test set but also on each part of the sub test set. The results justify the effectiveness of our online model optimization method.

5 Related work

Most previous research on SMT training data is focused on parallel data collection. Some work tries to acquire parallel sentences from web (Nie et al. 1999; Resnik and Smith 2003; Chen et al. 2004). Others extract parallel sentences from comparable or non-parallel corpora (Munteanu and Marcu 2005, 2006). These work aims to collect more parallel training corpora, while our work aims to make better use of existing parallel corpora.

Some research has been conducted on parallel data selection and adaptation. Eck et al. (2005) propose a method to select more informative sentences based on n-gram coverage. They use n-grams to estimate the importance of a sentence. The more previously unseen n-grams in the sentence the more important the sentence is. TF-IDF weighting scheme is also tried in their method, but didn't show improvements over n-grams. This method is independent of test data. Their goal is to decrease the amount of training data to make SMT system adaptable to small devices. Similar to our work, Hildebrand et al. (2005) also use information retrieval method for translation model adaptation. They select sentences similar to the test set from available in-of-domain and out-of-domain training data to form an adapted translation model. Different from their work, our method further use the small adapted data to optimize the distribution of the whole training data. It takes the full advantage of larger data and adapted data. In addition, we also propose an online translation model optimization method, which make it possible to select adapted translation model for each individual sentence.

Since large scale monolingual corpora are easier to obtain than parallel corpora. There has some research on language model adaptation recent years. Zhao et al. (2004) and Eck et al.(2004) introduce information retrieval method for language model adaptation. Zhang et al.(2006) and Mauser et al.(2006) use adapted language model for SMT re-ranking. Since language model is built for target language in SMT, one pass translation is usually needed to generate n-best translation candidates in language model adaptation. Translation model adaptation doesn't need a pre-translation procedure. Comparatively, it is more direct. Language model adaptation and translation model adaptation are good complement to each other. It is possible that combine these two adaptation approaches could further improve machine translation performance.

6 Conclusion and future work

This paper presents two new methods to improve statistical machine translation performance by making better use of the available parallel training corpora. The offline data selection method adapts the training corpora to the test domain by retrieving similar sentence pairs and redistributing their weight in the training data. Experimental results show that the selected small subset achieves comparable or even better performance than the baseline system with much less training data. The optimized training data can further improve translation performance without using any additional resource. The online model optimization method adapts the translation model to the online test sentence by redistributing the weight of each predefined submodels. Preliminary results show the effectiveness of the method. Our work also demonstrates that in addition to larger training data, more relevant training data is also important for SMT model training.

In future work, we will improve our methods in several aspects. Currently, the similar sentence retrieval model and the weighting schemes are very simple. It might work better by trying other sophisticated similarity measure models or using some optimization algorithms to determine submodel's weights. Introducing language model optimization into our system might further improve translation performance.

Acknowledgement

This work was supported by National Natural Science Foundation of China, Contract No. 60603095 and 60573188.

References

Jisong Chen, Rowena Chau, Chung-Hsing Yeh 2004. *Discovering Parallel Text from the World Wide Web*. ACSW Frontiers 2004: 157-161

Stanley F. Chen and Joshua Goodman. 1998. *An Empirical Study of Smoothing Techniques for Language Modeling*. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

Matthias Eck, Stephan Vogel, and Alex Waibel 2004. *Language Model Adaptation for Statistical Machine Translation Based on Information Retrieval*. Proceedings of Fourth International Conference on Language Resources and Evaluation:327-330

Matthias Eck, Stephan Vogel, Alex Waibel 2005. *Low cost portability for statistical machine translation based on n-gram coverage*. MT Summit X: 227-234.

Zhongjun He, Yang Liu, Deyi Xiong, Hongxu Hou, and Qun Liu 2006. *ICT System Description for the 2006 TC-STAR Run#2 SLT Evaluation*. Proceedings of TC-STAR Workshop on Speech-to-Speech Translation: 63-68

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. *Statistical phrase-based translation*. Proceedings of HLT-NAACL 2003: 127–133.

Arne Mauser, Richard Zens, Evgeny Matusov, Sasa Hasan, Hermann Ney 2006. *The RWTH Statistical Machine Translation System for the IWSLT 2006 Evaluation*. Proceedings of International Workshop on Spoken Language Translation.:103-110

Dragos Stefan Munteanu and Daniel Marcu 2005. *Improving Machine Translation Performance by Exploiting Comparable Corpora*. Computational Linguistics, 31 (4): 477-504

Dragos Stefan Munteanu and Daniel Marcu 2006. *Extracting Parallel Sub-Sentential Fragments from Comparable Corpora*. ACL-2006: 81-88

Jian-Yun Nie, Michel Simard, Pierre Isabelle, Richard Durand 1999. *Cross-Language Information Retrieval based on Parallel Texts and Automatic Mining of Parallel Texts in the Web.* SIGIR-1999: 74-81

Franz Josef Och 2003. *Minimum Error Rate Training in Statistical Machine Translation.* ACL-2003:160-167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. *Bleu: a Method for Automatic Evaluation of Machine Translation*. ACL-2002: 311–318

Philip Resnik and Noah A. Smith 2003. *The Web as a Parallel Corpus.* Computational Linguistics 29(3): 349-380

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel 2005. *Adaptation of the Translation Model for Statistical Machine Translation based on Information Retrieval*. Proceedings of EAMT 2005: 133-142.

Richard Zens, Franz Josef Och, Hermann Ney 2002. *Phrase-Based Statistical Machine Translation*. Annual German Conference on AI, KI 2002, Vol. LNAI 2479: 18-32

Ying Zhang, Almut Silja Hildebrand, Stephan Vogel 2006. *Distributed Language Modeling for N-best List Re-ranking.* EMNLP-2006:216-223

Bing Zhao, Matthias Eck, Stephan Vogel 2004. *Language Model Adaptation for Statistical Machine Translation with structured query models*. COLING-2004

Topic Segmentation with Hybrid Document Indexing

Irina Matveeva
Department of Computer Science
University of Chicago
Chicago, IL 60637
matveeva@cs.uchicago.edu

Gina-Anne Levow
Department of Computer Science
University of Chicago
Chicago, IL 60637
levow@cs.uchicago.edu

Abstract

We present a domain-independent unsupervised topic segmentation approach based on hybrid document indexing. Lexical chains have been successfully employed to evaluate lexical cohesion of text segments and to predict topic boundaries. Our approach is based in the notion of semantic cohesion. It uses spectral embedding to estimate semantic association between content nouns over a span of multiple text segments. Our method significantly outperforms the baseline on the topic segmentation task and achieves performance comparable to state-of-the-art methods that incorporate domain specific information.

1 Introduction

The goal of topic segmentation is to discover story boundaries in the stream of text or audio recordings. Story is broadly defined as segment of text containing topically related sentences. In particular, the task may require segmenting a stream of broadcast news, addressed by the Topic Detection and Tracking (TDT) evaluation project (Wayne, 2000; Allan, 2002). In this case topically related sentences belong to the same news story. While we are considering TDT data sets in this paper, we would like to pose the problem more broadly and consider a domain-independent approach to topic segmentation.

Previous research on topic segmentation has shown that lexical coherence is a reliable indicator of topical relatedness. Therefore, many approaches have concentrated on different ways of estimating lexical coherence of text segments, such as semantic similarity between words (Kozima, 1993), similarity between blocks of text (Hearst, 1994), and adaptive language models (Beeferman et al., 1999). These approaches use word repetitions to evaluate coherence. Since the sentences covering the same story represent a coherent discourse segment, they typically contain the same or related words. Repeated words build lexical chains that are consequently used to estimate lexical coherence. This can be done either by analyzing the number of overlapping lexical chains (Hearst, 1994) or by building a short-range and long-range language model (Beeferman et al., 1999). More recently, topic segmentation with lexical chains has been successfully applied to segmentation of news stories, multi-party conversation and audio recordings (Galley et al., 2003).

When the task is to segment long streams of text containing stories which may continue at a later point in time, for example developing news stories, building of lexical chains becomes intricate. In addition, the word repetitions do not account for synonymy and semantic relatedness between words and therefore may not be able to discover coherence of segments with little word overlap.

Our approach aims at discovering semantic relatedness beyond word repetition. It is based on the notion of semantic cohesion rather than lexical cohesion. We propose to use a similarity metric between segments of text that takes into account semantic associations between words spanning a number of segments. This method approximates lexical chains by averaging the similarity to a number of previous text

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 351–359, Prague, June 2007. ©2007 Association for Computational Linguistics

segments and accounts for synonymy by using a hybrid document indexing scheme. Our text segmentation experiments show a significant performance improvement over the baseline.

The rest of the paper is organized as follows. Section 2 discusses hybrid indexing. Section 3 describes our segmentation algorithm. Section 5 reports the experimental results. We conclude in section 6.

2 Hybrid Document Indexing

For the topic segmentation task we would like to define a similarity measure that accounts for synonymy and semantic association between words. This similarity measure will be used to evaluate semantic cohesion between text units and the decrease in semantic cohesion will be used as an indicator of a story boundary. First, we develop a document representation which supports this similarity measure.

Capturing semantic relations between words in a document representation is difficult. Different approaches tried to overcome the term independence assumption of the bag-of-words representation (Salton and McGill, 1983) by using distributional term clusters (Slonim and Tishby, 2000) and expanding the document vectors with synonyms, see (Levow et al., 2005). Since content words can be combined into semantic classes there has been a considerable interest in low-dimensional representations. Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is one of the best known dimensionality reduction algorithms. In the LSA space documents are indexed with latent semantic concepts. LSA maps all words to low dimensional vectors. However, the notion of semantic relatedness is defined differently for subsets of the vocabulary. In addition, the numerical information, abbreviations and the documents' style may be very good indicators of their topic. However, this information is no longer available after the dimensionality reduction.

We use a hybrid approach to document indexing to address these issues. We keep the notion of latent semantic concepts and also try to preserve the specifics of the document collection. Therefore, we divide the vocabulary into two sets: nouns and the rest of the vocabulary. The set of nouns does not include proper nouns. We use a method of spectral embedding, as described below and compute a low-dimensional representation for documents using only the nouns. We also compute a *tf-idf* representation for documents using the other set of words. Since we can treat each latent semantic concept in the low-dimensional representation as part of the vocabulary, we combine the two vector representations for each document by concatenating them.

2.1 Spectral Embedding

A vector space representation for documents and sentences is convenient and makes the similarity metrics such as cosine and distance readily available. However, those metrics will not work if they don't have a meaningful linguistic interpretation.

Spectral methods comprise a family of algorithms that embed terms and documents in a low-dimensional vector space. These methods use pairwise relations between the data points encoded in a similarity matrix. The main step is to find an embedding for the data that preserves the original similarities.

GLSA We use Generalized Latent Semantic Analysis (GLSA) (Matveeva et al., 2005) to compute spectral embedding for nouns. GLSA computes term vectors and since we would like to use spectral embedding for nouns, it is well-suited for our approach. GLSA extends the ideas of LSA by defining different ways to obtain the similarities matrix and has been shown to outperform LSA on a number of applications (Matveeva and Levow, 2006).

GLSA begins with a matrix of pair-wise term similarities S, computes its eigenvectors U and uses the first k of them to represent terms and documents, for details see (Matveeva et al., 2005). The justification for this approach is the theorem by Eckart and Young (Golub and Reinsch, 1971) stating that inner product similarities between the term vectors based on the eigenvectors of S represent the best element-wise approximation to the entries in S. In other words, the inner product similarity in the GLSA space preserves the semantic similarities in S.

Since our representation will try to preserve semantic similarities in S it is important to have a matrix of similarities which is linguistically motivated.

Word	Nearest Neighbors in GLSA Space					
witness	testify	prosecutor	trial	testimony	juror	eyewitness
finance	fund	bank	investment	economy	crisis	category
broadcast	television	TV	satellite	ABC	CBS	radio
hearing	hearing	judge	voice	chatter	sound	appeal
surprise	announcement	disappointment	stunning	shock	reaction	astonishment
rest	stay	remain	keep	leave	portion	economy

Table 1: Words' nearest neighbors in the GLSA semantic space.

2.2 Distributional Term Similarity

PMI Following (Turney, 2001; Matveeva et al., 2005), we use point-wise mutual information (PMI) to compute the matrix S. PMI between random variables representing the words w_i and w_j is computed as

$$PMI(w_i, w_j) = \log \frac{P(W_i = 1, W_j = 1)}{P(W_i = 1)P(W_j = 1)}. \quad (1)$$

Thus, for GLSA, $S(w_i, w_j) = PMI(w_i, w_j)$.

Co-occurrence Proximity An advantage of PMI is the notion of proximity. The co-occurrence statistics for PMI are typically computed using a sliding window. Thus, PMI will be large only for words that co-occur within a small context of fixed size.

Semantic Association vs. Synonymy Although GLSA was successfully applied to synonymy induction (Matveeva et al., 2005), we would like to point out that the GLSA discovers semantic association in a broad sense. Table 1 shows a few words from the TDT2 corpus and their nearest neighbors in the GLSA space. We can see that for "witness", "finance" and "broadcast" words are grouped into corresponding semantic classes. The nearest neighbors for "hearing" and "stay" represent their different senses. Interestingly, even for the abstract noun "surprise" the nearest neighbors are meaningful.

2.3 Document Indexing

We have two sets of the vocabulary terms: a set of nouns, N, and the other words, T. We compute *tf-idf* document vectors indexed with the words in T:

$$\vec{d_i} = (\alpha_i(w_1), \alpha_i(w_2), ..., \alpha_i(w_{|T|})), \quad (2)$$

where $\alpha_i(w_t) = \mathrm{tf}(w_t, d_i) * \mathrm{idf}(w_t)$.

We also compute a k-dimensional representation with latent concepts c_i as a weighted linear combination of GLSA term vectors $\vec{w_t}$:

$$\vec{d_i} = (c_1, ..., c_k) = \sum_{t=1:|N|} \alpha_i(w_t) * \vec{w_t}, \quad (3)$$

We concatenate these two representations to generate a hybrid indexing of documents:

$$\vec{d_i} = (\alpha_i(w_1), ..., \alpha_i(w_{|T|}), c_1, ...c_k) \quad (4)$$

In our experiments, we compute document and sentence representation using three indexing schemes: the *tf-idf* baseline, the GLSA representation and the hybrid indexing. The GLSA indexing computes term vectors for all vocabulary words; document and sentence vectors are generated as linear combinations of term vectors, as shown above.

2.4 Document similarity

One can define document similarity at different levels of semantic content. Documents can be similar because they discuss the same people or events and because they discuss related subjects and contain semantically related words. Hybrid Indexing allows us to combine both definitions of similarity. Each representation supports a different similarity measure. *tf-idf* uses term-matching, the GLSA representation uses semantic association in the latent semantic space computed for all words, and hybrid indexing uses a combination of both: term-matching for named entities and content words other than nouns combined with semantic association for nouns.

In the GLSA space, the inner product between document vectors contains all pair-wise inner product between their words, which allows one to detect semantic similarity beyond term matching:

$$\langle \vec{d_i}, \vec{d_j} \rangle = \sum_{w \in d_i} \sum_{v \in d_j} \alpha_i(w)\alpha_j(v)\langle \vec{w}, \vec{v} \rangle \quad (5)$$

If documents contain words which are different but semantically related, the inner product between the term vectors will contribute to the document similarity, as illustrated with an example in section 5.

When we compare two documents indexed with the hybrid indexing scheme, we compute a combination of similarity measures:

$$\langle \vec{d_i}, \vec{d_j} \rangle = \sum_{n_k \in d_i} \sum_{n_m \in d_j} \alpha_i(n_k)\alpha_j(n_m)\langle \vec{n_k}, \vec{n_m} \rangle + \sum_{t \in T} \alpha_i(t) * \alpha_j(t). \tag{6}$$

Document similarity contains semantic association between all pairs of nouns and uses term-matching for the rest of the vocabulary.

3 Topic Segmentation with Semantic Cohesion

Our approach to topic segmentation is based on semantic cohesion supported by the hybrid indexing. Topic segmentation approaches use either sentences (Galley et al., 2003) or blocks of words as text units (Hearst, 1994). We used both variants in our experiments. When using blocks, we computed blocks of a fixed size (typically 20 words) sliding over the documents in a fixed step size (10 or 5 words). The algorithm predicts a story boundary when the semantic cohesion between two consecutive units drops. Blocks can cross story boundaries, thus many predicted boundaries will be displaced with respect to the actual boundary.

Averaged similarity In our preliminary experiments we used the largest difference in score to predict story boundary, following the TextTiling approach (Hearst, 1994). We found, however, that in our document collection the word overlap between sentences was often not large and pair-wise similarity could drop to zero even for sentences within the same story, as will be illustrated below. We could not obtain satisfactory results with this approach.

Therefore, we used the average similarity by using a history of fixed size n. The semantic cohesion score was computed for the position between two text units, t_i and t_j as follows:

$$\text{score}(t_i, t_j) = \frac{1}{n} \sum_{k=0}^{n-1} \langle t_{i-k}, t_j \rangle \tag{7}$$

Our approach predicts story boundaries at the minima of the semantic cohesion score.

Approximating Lexical Chains One of the motivations for our cohesion score is that it approximates lexical chains, as for example in (Galley et al., 2003). Galley et al. (Galley et al., 2003) define lexical chains $R_1, .., R_N$ by considering repetitions of terms $t_1, .., t_N$ and assigning larger weights to short and compact chains. Then the lexical cohesion score between two text units t_i and t_j is based on the number of chains that overlap both of them:

$$\text{score}(t_i, t_j) = \sum_{k=1}^{N} w_k(t_i) w_k(t_j), \tag{8}$$

where $w_k(t_i) = \text{score}(R_j)$ if the chain R_j overlaps t_i and zero otherwise. Our cohesion score takes into account only the chains for words that occur in t_j and have another occurrence within n previous sentences. Due to this simplification, we compute the score based on inner products. Once we make the transition to inner products, we can use hybrid indexing and compute semantic cohesion score beyond term repetition.

4 Related Approaches

We compare our approach to the LCseg algorithm which uses lexical chains to estimate topic boundaries (Galley et al., 2003). Hybrid indexing allows us to compute semantic cohesion score rather than the lexical cohesion score based on word repetitions.

Choi at al. used LSA for segmentation (Choi et al., 2001). LSA (Deerwester et al., 1990) is a special case of spectral embedding and Choi at al. (Choi et al., 2001) used all vocabulary words to compute low-dimensional document vectors. We use GLSA (Matveeva et al., 2005) because it computes term vectors as opposed to the dual document-term representation with LSA and uses a different matrix of pair-wise similarities. Furthermore, Choi at al. (Choi et al., 2001) used clustering to predict boundaries whereas we used the average similarity scores.

s1: The **Cuban** news agency Prensa Latina called Clinton 's announcement Friday that Cubans picked up
at sea will be taken to Guantanamo Bay naval base a " new and dangerous element " in U S immigration policy.
s2: The **Cuban** government has not yet publicly reacted to Clinton 's announcement that Cuban rafters
will be turned away from the United States and taken to the U S base on the southeast tip of Cuba.
s5: The arrival of **Cuban** emigrants could be an " extraordinary aggravation " to the situation , Prensa Latina said.
s6: It noted that **Cuba** had already denounced the use of the base as a camp for Haitian refugees.
whom it had for many years encouraged to come to the United States.
s8: **Cuba** considers the land at the naval base , leased to the United States at the turn of the century,
to be illegally occupied.

s10: **General Motors Corp** said Friday it was recalling 5,600 1993-94 model Chevrolet Lumina, Pontiac
Trans Sport and Oldsmobile Silhouette minivans equipped with a power sliding door and built-in child seats.
s14: If this occurs , the shoulder belt may not properly retract , the *carmaker* said.
s15: **GM** is the only company to offer the power-sliding door.
s16: The *company* said it was not aware of any accidents or injuries related to the defect.
s17: To correct the problem , **GM** said dealers will install a modified interior trim piece that will reroute the seat belt.

Table 2: TDT. The first 17 sentences in the first file.

Existing approaches to hybrid indexing used different weights for proper nouns, nouns phrase heads and use WordNet synonyms to expand the documents, for example (Hatzivassiloglou et al., 2000; Hatzivassiloglou et al., 2001). Our approach does not require linguistic resources and learning the weights. The semantic associations between nouns are estimated using spectral embedding.

5 Experiments

5.1 Data

The first TDT collection is part of the LCseg toolkit[1] (Galley et al., 2003) and we used it to compare our approach to LCseg. We used the part of this collection with 50 files with 22 documents each.

We also used the TDT2 collection[2] of news articles from six news agencies in 1998. We used only 9,738 documents that are assigned to one topic and have length more than 50 words. We used the Lemur toolkit[3] with stemming and stop words list for the *tf-idf* indexing; we used Bikel's parser[4] to obtain the POS-tags and select nouns; we used the PLAPACK package (Bientinesi et al., 2003) to compute the eigenvalue decomposition.

Evaluation For the TDT data we use the error metric p_k (Beeferman et al., 1999) and WindowDiff (Pevzner and Hearst, 2002) which are implemented in the LCseg toolkit. We also used the TDT cost metric Cseg[5], with the default parameters P(seg)=0.3, Cmiss=1, Cfa=0.3 and distance of 50 words. All these measures look at two units (words or sentences) N units apart and evaluate how well the algorithm can predict whether there is a boundary between them or not. Lower values mean better performance for all measures.

Global vs. Local GLSA Similarity To obtain the PMI values we used the TDT2 collection, denoted as $GLSA_{local}$. Since co-occurrence statistics based on larger collections give a better approximation to linguistic similarities, we also used 700,000 documents from the English GigaWord collection, denoted as GLSA. We used a window of size 8.

5.2 Topic Segmentation

The first set of experiments was designed to evaluate the advantage of the GLSA representation over the baseline. We compare our approach to the LCseg algorithm (Galley et al., 2003) and use sentences as segmentation unit. To avoid the issue of parameters setting when the number of boundaries is not known, we provide each algorithm with the actual numbers

[1] http://www1.cs.columbia.edu/ galley/tools.html

[2] http://nist.gov/speech/tests/tdt/tdt98/

[3] http://www.lemurproject.org/

[4] http://www.cis.upenn.edu/ dbikel/software.html

[5] www.nist.gov/speech/tests/tdt/tdt98/doc/tdt2.eval.plan.98.v3.7.ps

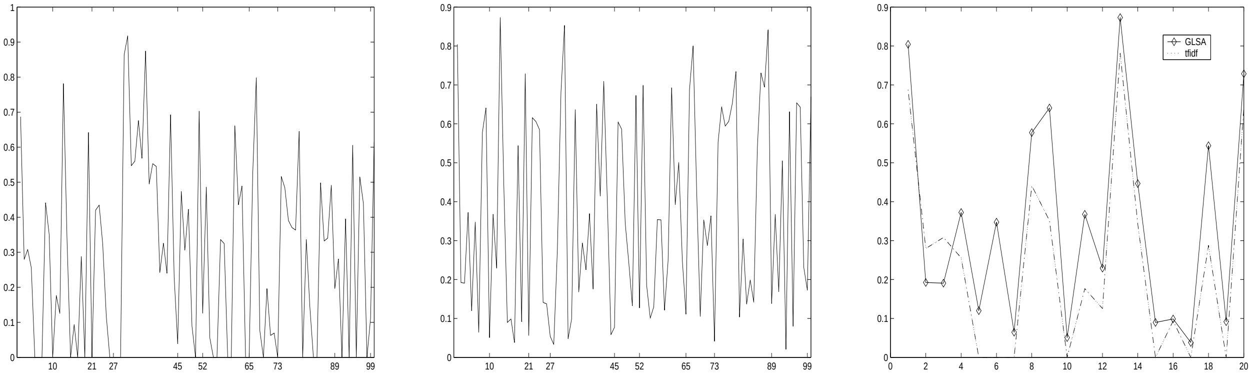


Figure 1: TDT. Pair-wise sentence similarities for *tf-idf* (left), GLSA (middle); x-axis shows story boundaries. Details for the first 20 sentences, table 2 (right).

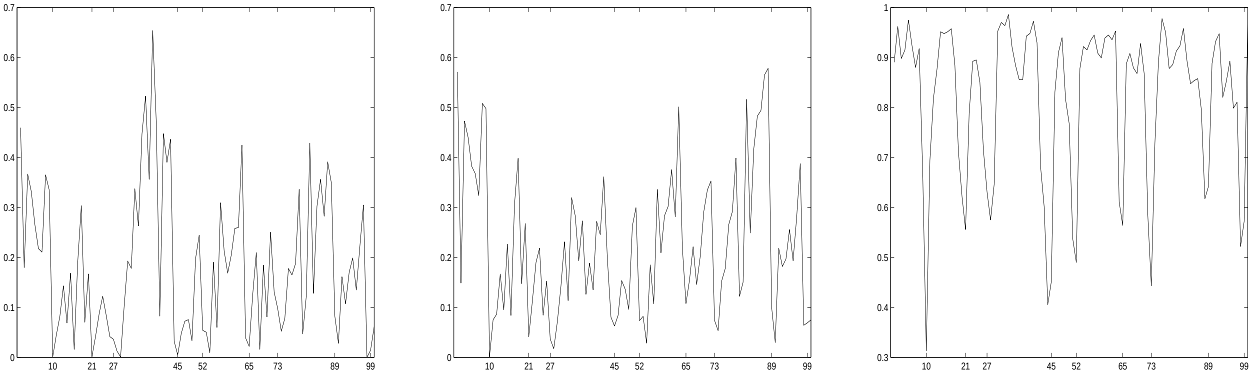


Figure 2: TDT. Pair-wise sentence similarities for *tf-idf* (left), GLSA (middle) averaged over 10 preceeding sentences; LCseg lexical cohesion scores (right). X-axis shows story boundaries.

of boundaries.

TDT We use the LCseg approach and our approach with the baseline *tf-idf* representation and the GLSA representation to segment this corpus. Table 2 shows a few sentences. Many content words are repeated, so the lexical chains is definitely a sound approach. As shown in Table 2, in the first story the word “Cuba” or “Cuban” is repeated in every sentence thus generating a lexical chain. On the topic boundary, the word overlap between sentences is very small. At the same time, the repetition of words may also be interrupted within a story: sentence 5, 6 and sentences 14, 15, 16 have little word overlap. LCseg deals with this by defining several parameters to control chain length and gaps. This simple example illustrates the potential benefit of semantic cohesion. Table 2 shows that “General Motors” or “GM” are not repeated in every sentence of the second story. However, “GM”, “carmaker” and “company” are semantically related. Making this information available to the segmentation algorithm allows it to establish a connection between each sentence of the second story.

We computed pair-wise sentence similarities between pairs of consecutive sentences in the *tf-idf* and GLSA representations. Figure 1 shows the similarity values plotted for each sentence break. The pairwise similarities based on term-matching are very spiky and there are many zeros within the story. The GLSA-based similarity makes the dips in the similarities at the boundaries more prominent. The last plot gives the details for the sentences in table 2. In the *tf-idf* representation sentences without word overlap receive zero similarity but the GLSA representation is able to use the semantic association between between “emigrants” and “refugees” for sentences 5 and 6, and also the semantic association between “carmaker” and “company” for sentences 14

Measure	*tf-idf*	GLSA	LCseg
Pmiss	0.29	0.19	N/A
Pfa	0.14	0.09	N/A
Cseg	0.18	0.08	N/A
p$_k$	0.24	0.17	0.07
wd	0.27	0.21	0.10

Table 3: TDT segmentation results.

and 15.

This effect increases as we use the semantic cohesion score as in equation 7. Figure 2 shows the similarity values for *tf-idf* and GLSA and also the lexical cohesion scores computed by LCseg. The GLSA-based similarities are not quite as smooth as the LCseg scores, but they correctly discover the boundaries. LCseg parameters are fine-tuned for this document collection. We used a general TDT2 GLSA representation for this collection, and the only segmentation parameter we used is to avoid placing next boundary within n=3 sentences of the previous one. For this reason the predicted boundary may be one sentence off the actual boundary. These results are summarized in Table 3. The GLSA representation performs significantly better than the *tf-idf* baseline. Its p_k and WindowDiff scores with default parameters for LCseg are worse than for LCseg. We attribute it to the fact that we did not fine-tuned our method to this collection and that boundaries are often placed one position off the actual boundary.

TDT2 For this collection we used three different indexing schemes: the *tf-idf* baseline, the GLSA representation and the hybrid indexing. Each representation supports a different similarity measure. Our TDT experiments showed that the semantic cohesion score based on the GLSA representation improves the segmentation results. The variant of the TDT corpus we used is rather small and well-balanced, see (Galley et al., 2003) for details. In the second phase of experiments we evaluate our approach on the larger TDT2 corpus. The experiments were designed to address the following issues:

- performance comparison between GLSA and Hybrid indexing representations. As mentioned before, GLSA embeds all words in a low-dimensional space. Whereas semantic classes for nouns have theoretical linguistic justification, it is harder to motivate a latent space representation for example for proper nouns. Therefore, we want to evaluate the advantage of using spectral embedding only for nouns.

- collection dependence of similarities. The similarity matrix S is computed using the TDT2 corpus ($GLSA_{local}$) and using the larger GigaWord corpus. The larger corpus provides more reliable co-occurrence statistics. On the other hand, word distribution is different from that in the TDT2 corpus. We wanted to evaluate whether semantic similarities are collection independent.

#b known			
Method	Pmiss	Pfa	Cseg
tf-idf	0.52	0.14	0.19
GLSA	0.4	0.1	0.14
GLSA_$local$	0.44	0.12	0.16
Hybrid	0.34	0.10	0.12
Hybrid_$local$	0.38	0.09	0.13
LCseg	0.80	0.19	0.28
#b unknown			
Method	Pmiss	Pfa	Cseg
tf-idf	0.42	0.2	0.17
GLSA	0.37	0.13	0.14
GLSA_$local$	0.35	0.19	0.14
Hybrid	0.26	0.16	0.11
Hybrid_$local$	0.27	0.18	0.12

Table 4: TDT2 segmentation results. Sliding blocks with size 20 and stepsize 10; similarity averaged over 10 preceeding blocks.

Table 4 shows the performance evaluation. We show the results computed using blocks containing 20 words (after preprocessing) with step size 10. We tried other parameter values but did not achieve better performance, which is consistent with other research (Hearst, 1994; Galley et al., 2003). We show the results for two settings: predict a known number of boundaries, and predict boundaries using a threshold. In our experiments we used the average of the smallest N scores as threshold, $N = 4000$ showing best results.

The spectral embedding based representations (GLSA, Hybrid) significantly outperform the baseline. This confirms the advantage of the semantic cohesion score vs. term-matching. Hybrid indexing outperforms the GLSA representation supporting our intuition that semantic association is best defined for nouns.

We used the GigaWord corpus to obtain the pair-wise word associations for the GLSA and Hybrid representations. We also computed GLSA$_{local}$ and Hybrid$_{local}$ using the TDT2 corpus to obtain the pair-wise word associations. The co-occurrence statistics based on the GigaWord corpus provide more reliable estimations of semantic association despite the difference in term distribution. The difference is larger for the GLSA case when we compute the embedding for all words, GLSA performs better than GLSA$_{local}$. Hybrid$_{local}$ performs only slightly worse than Hybrid. This seems to support the claim that semantic associations between nouns are largely collection independent. On the other hand, semantic associations for proper names are collection dependent at least because the collections are static but the semantic relations of proper names may change over time. The semantic space for a name of a president, for example, is different for the period of time of his presidency and for the time before and after that.

Disappointingly, we could not achieve good results with LCseg. It tends to split stories into short paragraphs. Hybrid indexing could achieve results comparable to state-of-the art approaches, see (Fiscus et al., 1998) for an overview.

6 Conclusion and Future Work

We presented a topic segmentation approach based on semantic cohesion scores. Our approach is domain independent, does not require training or use of lexical resources. The scores are computed based on the hybrid document indexing which uses spectral embedding in the space of latent concepts for nouns and keeps proper nouns and other specifics of the documents collections unchanged. We approximate the lexical chains approach by simplifying the definition of a chain which allows us to use inner products as basis for the similarity score. The similarity score takes into account semantic relations between nouns beyond term matching. This semantic cohesion approach showed good results on the topic segmentation task.

We intend to extend the hybrid indexing approach by considering more vocabulary subsets. Syntactic similarity is more appropriate for verbs, for example, than co-occurrence. As a next step, we intend to embed verbs using syntactic similarity. It would also be interesting to use lexical chains for proper names and learn the weights for different similarity scores.

References

J. Allan, editor. 2002. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers.

Doug Beeferman, Adam Berger, and John D. Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210.

Paolo Bientinesi, Inderjit S. Dhilon, and Robert A. van de Geijn. 2003. A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations. *UT CS Technical Report TR-03-26*.

Freddy Choi, Peter Wiemer-Hastings, and Johanna Moore. 2001. Latent Semantic Analysis for text segmentation. In *Proceedings of EMNLP*, pages 109–117.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

J. G. Fiscus, George Doddington, John S. Garofolo, and Alvin Martin. 1998. NIST's 1998 topic detection and tracking evaluation (tdt2). In *Proceedings of NIST's 1998 Topic Detection and Tracking Evaluation*.

M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of ACL*.

G. Golub and C. Reinsch. 1971. *Handbook for Matrix Computation II, Linear Algebra*. Springer-Verlag, New York.

V. Hatzivassiloglou, Luis Gravano, and Ankineedu Maganti. 2000. An investigation of linguistic features and clustering algorithms for topical document clustering. In *Proceedings of SIGIR*, pages 224–231.

V. Hatzivassiloglou, Regina Barzilay Min-Yen Kan Judith L. Klavans, Melissa L. Holcombe, and Kathleen R. McKeown. 2001. Simfinder: A flexible clustering tool for summarization. In *Proceedings of NAACL*, pages 41–49.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL*, pages 9–16.

Hideki Kozima. 1993. Text segmentation based on similarity between words. In *Proceedings of ACL*, pages 286–288.

Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. *Information Processing and Management: Special Issue on Cross-language Information Retrieval*.

Irina Matveeva and Gina-Anne Levow. 2006. Graph-based Generalized Latent Semantic Analysis for document representation. In *Proc. of the TextGraphs Workshop at HLT/NAACL*.

Irina Matveeva, Gina-Anne Levow, Ayman Farahat, and Christian Royer. 2005. Generalized Latent Semantic Analysis for term representation. In *Proc. of RANLP*.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Comput. Linguist.*, 28(1):19–36.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Noam Slonim and Naftali Tishby. 2000. Document clustering using word clusters via the information bottleneck method. In *Research and Development in Information Retrieval*, pages 208–215.

Peter D. Turney. 2001. Mining the web for synonyms: PMI–IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167:491–502.

C. Wayne. 2000. Multilingual topic detection and tracking: Successful research enabled by corpora and evaluation. In *Proceedings of Language Resources and Evaluation Conference (LREC)*, pages 1487–1494.

Syntactic Re-Alignment Models for Machine Translation

Jonathan May
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
jonmay@isi.edu

Kevin Knight
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
knight@isi.edu

Abstract

We present a method for improving word alignment for statistical syntax-based machine translation that employs a syntactically informed alignment model closer to the translation model than commonly-used word alignment models. This leads to extraction of more useful linguistic patterns and improved BLEU scores on translation experiments in Chinese and Arabic.

1 Methods of statistical MT

Roughly speaking, there are two paths commonly taken in statistical machine translation (Figure 1). The idealistic path uses an unsupervised learning algorithm such as EM (Demptser et al., 1977) to learn parameters for some proposed translation model from a bitext training corpus, and then directly translates using the weighted model. Some examples of the idealistic approach are the direct IBM word model (Berger et al., 1994; Germann et al., 2001), the phrase-based approach of Marcu and Wong (2002), and the syntax approaches of Wu (1996) and Yamada and Knight (2001). Idealistic approaches are conceptually simple and thus easy to relate to observed phenomena. However, as more parameters are added to the model the idealistic approach has not scaled well, for it is increasingly difficult to incorporate large amounts of training data efficiently over an increasingly large search space. Additionally, the EM procedure has a tendency to overfit its training data when the input units have varying explanatory powers, such as variable-size phrases or variable-height trees.

The realistic path also learns a model of translation, but uses that model only to obtain Viterbi word-for-word alignments for the training corpus. The bitext and corresponding alignments are then used as input to a pattern extraction algorithm, which yields a set of patterns or rules for a second translation model (which often has a wider parameter space than that used to obtain the word-for-word alignments). Weights for the second model are then set, typically by counting and smoothing, and this weighted model is used for translation. Realistic approaches scale to large data sets and have yielded better BLEU performance than their idealistic counterparts, but there is a disconnect between the first model (hereafter, the *alignment model*) and the second (the *translation model*). Examples of realistic systems are the phrase-based ATS system of Och and Ney (2004), the phrasal-syntax hybrid system Hiero (Chiang, 2005), and the GHKM syntax system (Galley et al., 2004; Galley et al., 2006). For an alignment model, most of these use the Aachen HMM approach (Vogel et al., 1996), the implementation of IBM Model 4 in GIZA++ (Och and Ney, 2000) or, more recently, the semi-supervised EMD algorithm (Fraser and Marcu, 2006).

The two-model approach of the realistic path has undeniable empirical advantages and scales to large data sets, but new research tends to focus on development of higher order translation models that are informed only by low-order alignments. We would like to add the analytic power gained from modern translation models to the underlying alignment model without sacrificing the efficiency and empirical gains of the two-model approach. By adding the

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 360–368, Prague, June 2007. ©2007 Association for Computational Linguistics

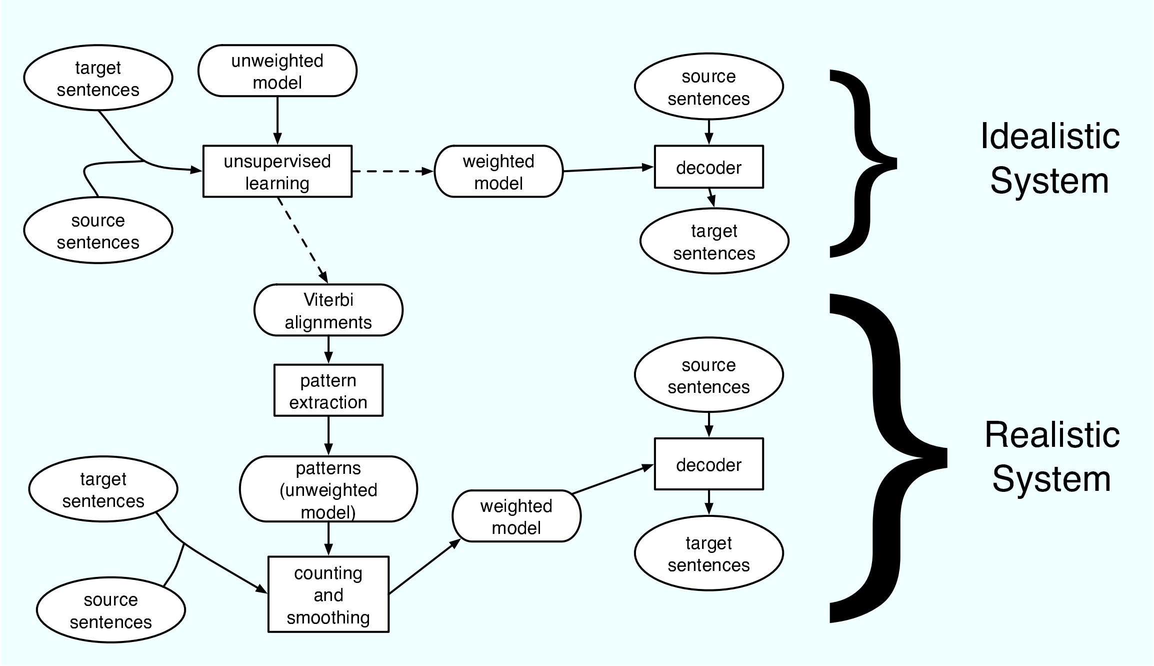


Figure 1: General approach to idealistic and realistic statistical MT systems

syntactic information used in the translation model to our alignment model we may improve alignment quality such that rule quality and, in turn, system quality are improved. In the remainder of this work we show how a touch of idealism can improve an existing realistic syntax-based translation system.

2 Multi-level syntactic rules for syntax MT

Galley et al. (2004) and Galley et al. (2006) describe a syntactic translation model that relates English trees to foreign strings. The model describes joint production of a (tree, string) pair via a non-deterministic selection of weighted rules. Each rule has an English tree fragment with variables and a corresponding foreign string fragment with the same variables. A series of rules forms an explanation (or *derivation*) of the complete pair.

As an example, consider the parsed English and corresponding Chinese at the top of Figure 2. The three columns underneath the example are different rule sequences that can explain this pair; there are many other possibilities. Note how rules specify rotation (e.g. R10, R5), direct translation (R12, R8), insertion and deletion (R11, R1), and tree traversal (R7, R15). Note too that the rules explain variable-size fragments (e.g. R6 vs. R14) and thus the possible *derivation trees* of rules that explain a sentence pair have varying sizes. The smallest such derivation tree has a single large rule (which does not appear in Figure 2; we leave the description of such a rule as an exercise for the reader). A string-to-tree decoder constructs a *derivation forest* of derivation trees where the right sides of the rules in a tree, taken together, explain a candidate source sentence. It then outputs the English tree corresponding to the highest-scoring derivation in the forest.

3 Introducing syntax into the alignment model

We now lay the ground for a syntactically motivated alignment model. We begin by reviewing an alignment model commonly seen in realistic MT systems and compare it to a syntactically-aware alignment model.

3.1 The traditional IBM alignment model

IBM Model 4 (Brown et al., 1993) learns a set of 4 probability tables to compute $p(f|e)$ given a foreign sentence f and its target translation e via the following (greatly simplified) generative story:

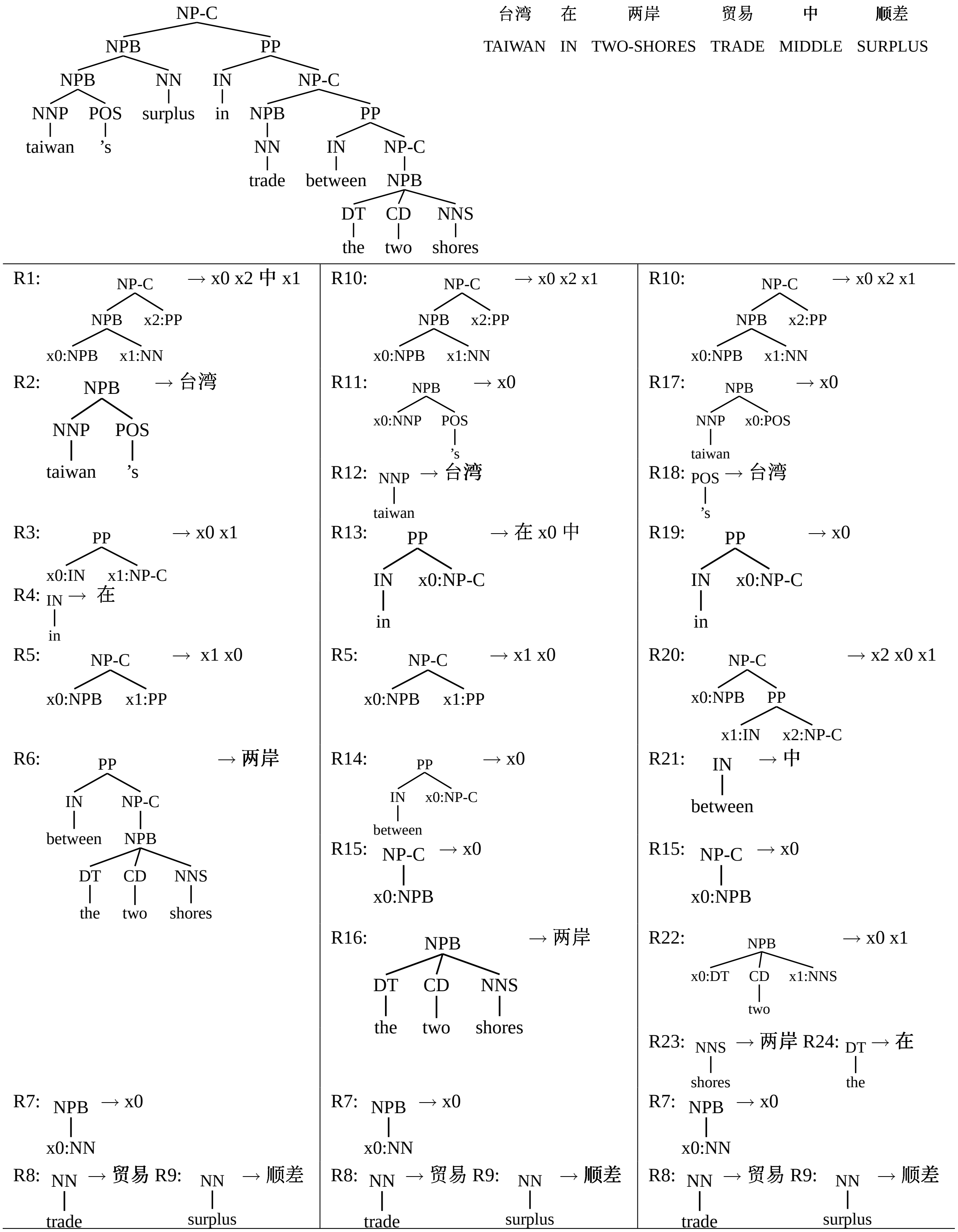


Figure 2: A (English tree, Chinese string) pair and three different sets of multilevel tree-to-string rules that can explain it; the first set is obtained from bootstrap alignments, the second from this paper’s re-alignment procedure, and the third is a viable, if poor quality, alternative that is not learned.

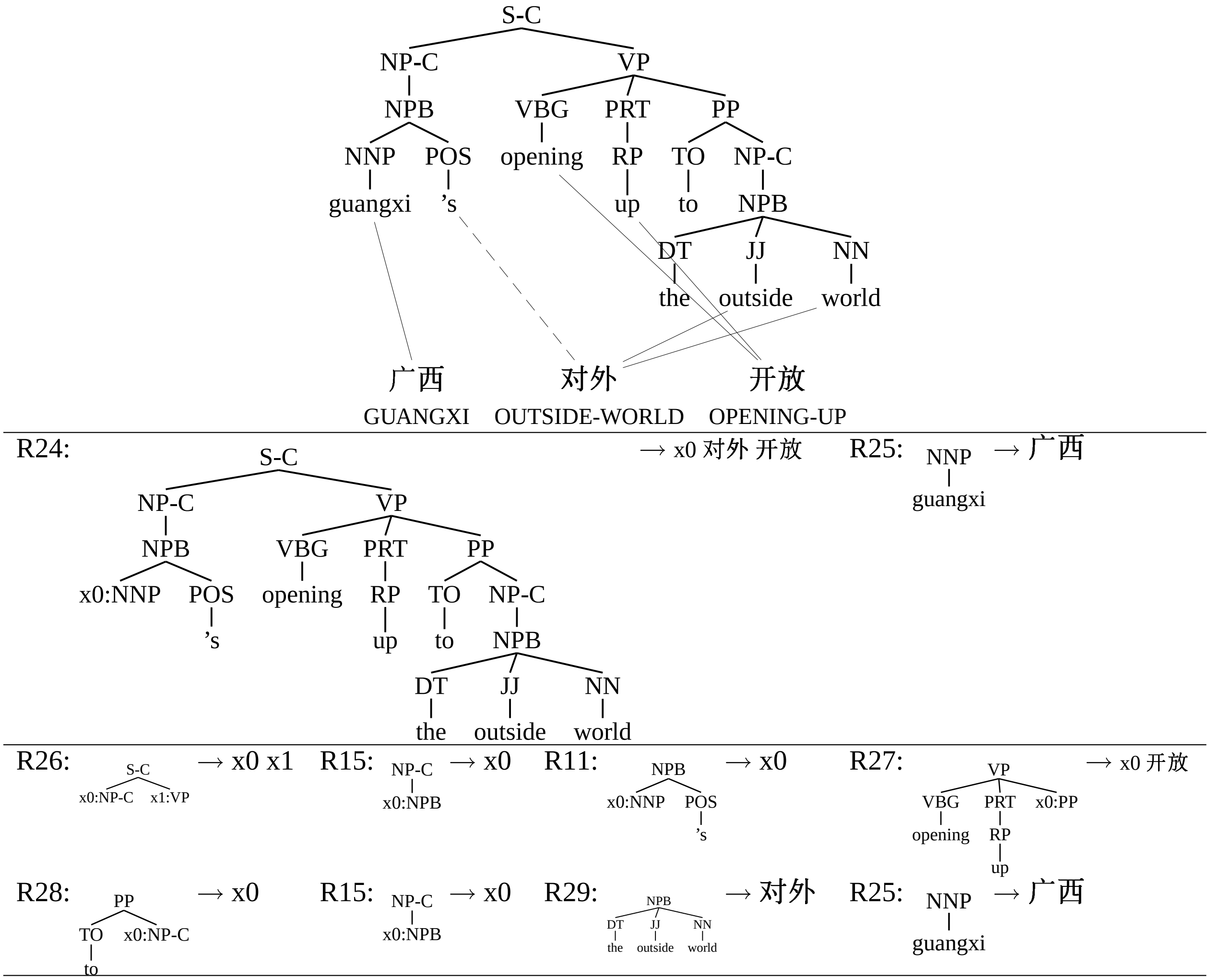


Figure 3: The impact of a bad alignment on rule extraction. Including the alignment link indicated by the dotted line in the example leads to the rule set in the second row. The re-alignment procedure described in Section 3.2 learns to prefer the rule set at bottom, which omits the bad link.

1. A fertility y for each word e_i in e is chosen with probability $p_{fert}(y|e_i)$.
2. A null word is inserted next to each fertility-expanded word with probability p_{null}.
3. Each token e_i in the fertility-expanded word and null string is translated into some foreign word f_i in f with probability $p_{trans}(f_i|e_i)$.
4. The position of each foreign word f_i that was translated from e_i is changed by Δ (which may be positive, negative, or zero) with probability $p_{distortion}(\Delta|\mathcal{A}(e_i), \mathcal{B}(f_i))$, where $\mathcal{A}$ and $\mathcal{B}$ are functions over the source and target vocabularies, respectively.

Brown et al. (1993) describes an EM algorithm for estimating values for the four tables in the generative story. However, searching the space of all possible alignments is intractable for EM, so in practice the procedure is bootstrapped by models with narrower search space such as IBM Model 1 (Brown et al., 1993) or Aachen HMM (Vogel et al., 1996).

3.2 A syntax re-alignment model

Now let us contrast this commonly used model for obtaining alignments with a syntactically motivated alternative. We recall the rules described in Section 2. Our model learns a single probability table to compute $p(etree, f)$ given a foreign sentence f and a parsed target translation $etree$. In the following generative story we assume a starting variable with syntactic type v.

1. Choose a rule r to replace v, with probability $p_{rule}(r|v)$.
2. For each variable with syntactic type v_i in the partially completed (tree, string) pair, continue to choose rules r_i with probability $p_{rule}(r_i|v_i)$ to replace these variables until there are no variables remaining.

In Section 5.1 we discuss an EM learning procedure for estimating these rule probabilities.

As in the IBM approach, we must mitigate intractability by limiting the parameter space searched, which is potentially much wider than in the word-to-word case. We would like to supply to EM all possible rules that explain the training data, but this implies a rule relating each possible tree fragment to each possible string fragment, which is infeasible. We follow the approach of bootstrapping from a model with a narrower parameter space as is done in, e.g. Och and Ney (2000) and Fraser and Marcu (2006).

To reduce the model space we employ the rule acquisition technique of Galley et al. (2004), which obtains rules given a (tree, string) pair as well as an initial alignment between them. We are agnostic about the source of this bootstrap alignment and in Section 5 present results based on several different bootstrap alignment qualities. We require an initial set of alignments, which we obtain from a word-for-word alignment procedure such as GIZA++ or EMD. Thus, we are not aligning input data, but rather *re-aligning* it with a syntax model.

4 The appeal of a syntax alignment model

Consider the example of Figure 2 again. The leftmost derivation is obtained from the bootstrap alignment set. This derivation is reasonable but there are some poorly motivated rules, from a linguistic standpoint. The Chinese word 两岸 roughly means "the two shores" in this context, but the rule R6 learned from the alignment incorrectly includes "between". However, other sentences in the training corpus have the correct alignment, which yields rule R16. Meanwhile, rules R13 and R14, learned from yet other sentences in the training corpus, handle the 在 ... 中 structure (which roughly translates to "in between"), thus allowing the middle derivation.

		SENTENCE PAIRS	
	DESCRIPTION	CHINESE	ARABIC
TUNE	NIST 2002 short	925	696
TEST	NIST 2003	919	663

Table 1: Tuning and testing data sets for the MT system described in Section 5.2.

EM distributes rule probabilities in such a way as to maximize the probability of the training corpus. It thus prefers to use one rule many times instead of several different rules for the same situation over several sentences, if possible. R6 is a possible rule in 46 of the 329,031 sentence pairs in the training corpus, while R16 is a possible rule in 100 sentence pairs. Well-formed rules are more usable than ill-formed rules and the partial alignments behind these rules, generally also well-formed, become favored as well. The top row of Figure 3 contains an example of an alignment learned by the bootstrap alignment model that includes an incorrect link. Rule R24, which is extracted from this alignment, is a poor rule. A set of commonly seen rules learned from other training sentences provide a more likely explanation of the data, and the consequent alignment omits the spurious link.

5 Experiments

In this section, we describe the implementation of our semi-idealistic model and our means of evaluating the resulting re-alignments in an MT task.

5.1 The re-alignment setup

We begin with a training corpus of Chinese-English and Arabic-English bitexts, the English side parsed by a reimplementation of the standard Collins model (Bikel, 2004). In order to acquire a syntactic rule set, we also need a bootstrap alignment of each training sentence. We use an implementation of the GHKM

BOOTSTRAP GIZA CORPUS		RE-ALIGNMENT EXPERIMENT			
ENGLISH WORDS	CHINESE WORDS	TYPE	RULES	TUNE	TEST
9,864,294	7,520,779	baseline	19,138,252	39.08	37.77
		initial	18,698,549	39.49	38.39
		adjusted	26,053,341	**39.76**	**38.69**

Table 2: A comparison of Chinese BLEU performance between the GIZA baseline (no re-alignment), re-alignment as proposed in Section 3.2, and re-alignment as modified in Section 5.4

algorithm (Galley et al., 2004) to obtain a rule set for each bootstrap alignment.

Now we need an EM algorithm for learning the parameters of the rule set that maximize $\prod_{corpus} p(tree, string)$. Such an algorithm is presented by Graehl and Knight (2004). The algorithm consists of two components: DERIV, which is a procedure for constructing a packed forest of derivation trees of rules that explain a (tree, string) bitext corpus given that corpus and a rule set, and TRAIN, which is an iterative parameter-setting procedure.

We initially attempted to use the top-down DERIV algorithm of Graehl and Knight (2004), but as the constraints of the derivation forests are largely lexical, too much time was spent on exploring dead-ends. Instead we build derivation forests using the following sequence of operations:

1. Binarize rules using the synchronous binarization algorithm for tree-to-string transducers described in Zhang et al. (2006).
2. Construct a parse chart with a CKY parser simultaneously constrained on the foreign string and English tree, similar to the bilingual parsing of Wu (1997) [1].
3. Recover all reachable edges by traversing the chart, starting from the topmost entry.

Since the chart is constructed bottom-up, leaf lexical constraints are encountered immediately, resulting in a narrower search space and faster running time than the top-down DERIV algorithm for this application. Derivation forest construction takes around 400 hours of cumulative machine time (4-processor machines) for Chinese. The actual running of EM iterations (which directly implements the TRAIN algorithm of Graehl and Knight (2004)) takes about 10 minutes, after which the Viterbi derivation trees are directly recoverable. The Viterbi derivation tree tells us which English words produce which Chinese words, so we can extract a word-to-word alignment from it. We summarize the approach described in this paper as:

1. Obtain bootstrap alignments for a training corpus using GIZA++.
2. Extract rules from the corpus and alignments using GHKM, noting the partial alignment that is used to extract each rule.
3. Construct derivation forests for each (tree, string) pair, ignoring the alignments, and run EM to obtain Viterbi derivation trees, then use the annotated partial alignments to obtain Viterbi alignments.
4. Use the new alignments as input to the MT system described below.

5.2 The MT system setup

A truly idealistic MT system would directly apply the rule weight parameters learned via EM to a machine translation task. As mentioned in Section 1, we maintain the two-model, or realistic approach. Below we briefly describe the translation model, focusing on comparison with the previously described alignment model. Galley et al. (2006) provides a more complete description of the translation model and DeNeefe et al. (2007) provides a more complete description of the end-to-end translation pipeline.

Although in principle the re-alignment model and translation model learn parameter weights over the same rule space, in practice we limit the rules used for re-alignment to the set of smallest rules that explain the training corpus and are consistent with the bootstrap alignments. This is a compromise made to reduce the search space for EM. The translation model learns multiple derivations of rules consistent with the re-alignments for each sentence, and learns

[1] In the cases where a rule is not synchronous-binarizable standard left-right binarization is performed and proper permutation of the disjoint English tree spans must be verified when building the part of the chart that uses this rule.

(a) Chinese re-alignment corpus has 9,864,294 English and 7,520,779 Chinese words

BOOTSTRAP GIZA CORPUS		RE-ALIGNMENT EXPERIMENT			
ENGLISH WORDS	CHINESE WORDS	TYPE	RULES	TUNE	TEST
9,864,294	7,520,779	baseline	19,138,252	39.08	37.77
		re-alignment	26,053,341	**39.76**	**38.69**
221,835,870	203,181,379	baseline	23,386,535	39.51	38.93
		re-alignment	33,374,646	**40.17**	**39.96**

(b) Arabic re-alignment corpus has 4,067,454 English and 3,147,420 Arabic words

BOOTSTRAP GIZA CORPUS		RE-ALIGNMENT EXPERIMENT			
ENGLISH WORDS	ARABIC WORDS	TYPE	RULES	TUNE	TEST
4,067,454	3,147,420	baseline	2,333,839	**47.92**	47.33
		re-alignment	2,474,737	47.87	**47.89**
168,255,347	147,165,003	baseline	3,245,499	49.72	49.60
		re-alignment	3,600,915	49.73	**49.99**

Table 3: Machine Translation experimental results evaluated with case-insensitive BLEU4.

weights for these by counting and smoothing. A dozen other features are also added to the rules. We obtain weights for the combinations of the features by performing minimum error rate training (Och, 2003) on held-out data. We then use a CKY decoder to translate unseen test data using the rules and tuned weights. Table 1 summarizes the data used in tuning and testing.

5.3 Initial results

An initial re-alignment experiment shows a reasonable rise in BLEU scores from the baseline (Table 2), but closer inspection of the rules favored by EM implies we can do even better. EM has a tendency to favor few large rules over many small rules, even when the small rules are more useful. Referring to the rules in Figure 2, note that possible derivations for (taiwan 's, 台湾)[2] are R2, R11-R12, and R17-R18. Clearly the third derivation is not desirable, and we do not discuss it further. Between the first two derivations, R11-R12 is preferred over R2, as the conditioning for possessive insertion is not related to the specific Chinese word being inserted. Of the 1,902 sentences in the training corpus where this pair is seen, the bootstrap alignments yield the R2 derivation 1,649 times and the R11-R12 derivation 0 times. Re-alignment does not change the result much; the new alignments yield the R2 derivation 1,613 times and again never choose R11-R12. The rules in the second derivation themselves are not rarely seen – R11 is in 13,311 forests *other* than those where R2 is seen, and R12 is in 2,500 additional forests. EM gives R11 a probability of $e^{-7.72}$ – better than 98.7% of rules, and R12 a probability of $e^{-2.96}$. But R2 receives a probability of $e^{-6.32}$ and is preferred over the R11-R12 derivation, which has a combined probability of $e^{-10.68}$.

5.4 Making EM fair

The preference for shorter derivations containing large rules over longer derivations containing small rules is due to a general tendency for EM to prefer derivations with few atoms. Marcu and Wong (2002) note this preference but consider the phenomenon a feature, rather than a bug. Zollmann and Sima'an (2005) combat the overfitting aspect for parsing by using a held-out corpus and a straight maximum likelihood estimate, rather than EM. We take a modeling approach to the phenomenon.

As the probability of a derivation is determined by the product of its atom probabilities, longer derivations with more probabilities to multiply have an inherent disadvantage against shorter derivations, all else being equal. EM is an iterative procedure and thus such a bias can lead the procedure to converge with artificially raised probabilities for short derivations and the large rules that comprise them. The relatively rare applicability of large rules (and thus lower observed partial counts) does not overcome the inherent advantage of large coverage. To combat this, we introduce size terms into our generative story, ensuring that all competing derivations for the

[2] The Chinese gloss is simply "taiwan".

LANGUAGE PAIR	TYPE	RULES	TUNE	TEST
CHINESE-ENGLISH	baseline	55,781,061	**41.51**	40.55
	EMD re-align	69,318,930	41.23	40.55
ARABIC-ENGLISH	baseline	8,487,656	**51.90**	51.69
	EMD re-align	11,498,150	51.88	**52.11**

Table 4: Re-alignment performance with semi-supervised EMD bootstrap alignments

same sentence contain the same number of atoms:

1. Choose a rule size s with cost $c_{size}(s)^{s-1}$.
2. Choose a rule r (of size s) to replace the start symbol with probability $p_{rule}(r|s, v)$.
3. For each variable in the partially completed (tree, string) pair, continue to choose sizes followed by rules, recursively to replace these variables until there are no variables remaining.

This generative story changes the derivation comparison from R2 vs R11-R12 to S2-R2 vs R11-R12, where S2 is the atom that represents the choice of size 2 (the size of a rule in this context is the number of non-leaf and non-root nodes in its tree fragment). Note that the variable number of inclusions implied by the exponent in the generative story above ensures that all derivations have the same size. For example, a derivation with one size-3 rule, a derivation with one size-2 and one size-1 rule, and a derivation with three size-1 rules would each have three atoms. With this revised model that allows for fair comparison of derivations, the R11-R12 derivation is chosen 1636 times, and S2-R2 is not chosen. R2 does, however, appear in the translation model, as the expanded rule extraction described in Section 5.2 creates R2 by joining R11 and R12.

The probability of size atoms, like that of rule atoms, is decided by EM. The revised generative story tends to encourage smaller sizes by virtue of the exponent. This does not, however, simply ensure the largest number of rules per derivation is used in all cases. Ill-fitting and poorly-motivated rules such as R22, R23, and R24 in Figure 2 are not preferred over R16, even though they are smaller. However, R14 and R16 are preferred over R6, as the former are useful rules. Although the modified model does not sum to 1, it leads to an improvement in BLEU score, as can be seen in the last row of Table 2.

5.5 Results

We performed primary experiments on two different bootstrap setups in two languages: the initial experiment uses the same data set for the GIZA++ initial alignment as is used in the re-alignment, while an experiment on better quality bootstrap alignments uses a much larger data set. For each bootstrapping in each language we compared the baseline of using these alignments directly in an MT system with the experiment of using the alignments obtained from the re-alignment procedure described in Section 5.4. For each experiment we report: the number of rules extracted by the expanded GHKM algorithm of Galley et al. (2006) for the translation model, converged BLEU scores on the tuning set, and finally BLEU performance on the held-out test set. Data set specifics for the GIZA++ bootstrapping and BLEU results are summarized in Table 3.

5.6 Discussion

The results presented demonstrate we are able to improve on unsupervised GIZA++ alignments by about 1 BLEU point for Chinese and around 0.4 BLEU point for Arabic using an additional unsupervised algorithm that requires no human aligned data. If human-aligned data is available, the EMD algorithm provides higher baseline alignments than GIZA++ that have led to better MT performance (Fraser and Marcu, 2006). As a further experiment we repeated the experimental conditions from Table 3, this time bootstrapped with the semi-supervised EMD method, which uses the larger bootstrap GIZA corpora described in Table 3 and an additional 64,469/48,650 words of hand-aligned English-Chinese and 43,782/31,457 words of hand-aligned English-Arabic. The results of this advanced experiment are in Table 4. We show a 0.42 gain in BLEU for Arabic, but no movement for Chinese. We believe increasing the size of the re-alignment corpora will increase BLEU gains in this experimental condition, but leave those results for future work.

We can see from the results presented that the impact of the syntax-aware re-alignment procedure of Section 3.2, coupled with the addition of size parameters to the generative story from Section 5.4 serves to remove links from the bootstrap alignments that cause less useful rules to be extracted, and thus increase the overall quality of the rules, and hence the system performance. We thus see the benefit to including syntax in an alignment model, bringing the two models of the realistic machine translation path somewhat closer together.

Acknowledgments

We thank David Chiang, Steve DeNeefe, Alex Fraser, Victoria Fossum, Jonathan Graehl, Liang Huang, Daniel Marcu, Michael Pust, Oana Postolache, Michael Pust, Jason Riesa, Jens Vöckler, and Wei Wang for help and discussion. This research was supported by NSF (grant IIS-0428020) and DARPA (contract HR0011-06-C-0022).

References

Adam Berger, Peter Brown, Stephen Della Pietra, Vincent Della Pietra, John Gillett, John Lafferty, Robert Mercer, Harry Printz, and Luboš Ureš. 1994. The candide system for machine translation. In *Proc. HLT*, pages 157–162, Plainsboro, New Jersey, March.

Daniel Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270, Ann Arbor, Michigan, June.

Arthur P. Demptser, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. EMNLP/CONLL*, Prague, June.

Alexander Fraser and Daniel Marcu. 2006. Semi-supervised training for statistical word alignment. In *Proc. COLING-ACL*, pages 769–776, Sydney, July.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL*, pages 273–280, Boston, May.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steven DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic models. In *Proc. COLING-ACL*, pages 961–968, Sydney, July.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proc. ACL*, pages 228–235, Toulouse, France, July.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proc. HLT-NAACL*, pages 105–112, Boston, May.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proc. EMNLP*, pages 133–139, Philadelphia, July.

Franz Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proc. ACL*, pages 440–447, Hong Kong, October.

Franz Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. ACL*, pages 160–167, Sapporo, Japan, July.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. COLING*, pages 836–841, Copenhagen, August.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. ACL*, pages 152–158, Santa Cruz, California, June.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. ACL*, pages 523–530, Toulouse, France, July.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. HLT-NAACL*, pages 256–263, New York City, June.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2/3):367–388.

Detecting Compositionality of Verb-Object Combinations using Selectional Preferences

Diana McCarthy
University of Sussex
Falmer, East Sussex
BN1 9QH, UK
dianam@sussex.ac.uk

Sriram Venkatapathy
International Institute
of Information Technology
Hyderabad, India
sriram@research.iiit.ac.in

Aravind K. Joshi
University of Pennsylvania,
Philadelphia
PA, USA.
joshi@linc.cis.upenn.edu

Abstract

In this paper we explore the use of selectional preferences for detecting non-compositional verb-object combinations. To characterise the arguments in a given grammatical relationship we experiment with three models of selectional preference. Two use WordNet and one uses the entries from a distributional thesaurus as classes for representation. In previous work on selectional preference acquisition, the classes used for representation are selected according to the coverage of argument tokens rather than being selected according to the coverage of argument types. In our distributional thesaurus models and one of the methods using WordNet we select classes for representing the preferences by virtue of the number of argument types that they cover, and then only tokens under these classes which are representative of the argument head data are used to estimate the probability distribution for the selectional preference model. We demonstrate a highly significant correlation between measures which use these 'type-based' selectional preferences and compositionality judgements from a data set used in previous research. The type-based models perform better than the models which use tokens for selecting the classes. Furthermore, the models which use the automatically acquired thesaurus entries produced the best results. The correlation for the thesaurus models is stronger than any of the individual features used in previous research on the same dataset.

1 Introduction

Characterising the semantic behaviour of phrases in terms of compositionality has particularly attracted attention in recent years (Lin, 1999; Schone and Jurafsky, 2001; Bannard, 2002; Bannard et al., 2003; Baldwin et al., 2003; McCarthy et al., 2003; Bannard, 2005; Venkatapathy and Joshi, 2005). Typically the phrases are putative multiwords and non-compositionality is viewed as an important feature of many such "words with spaces" (Sag et al., 2002). For applications such as paraphrasing, information extraction and translation, it is essential to take the words of non-compositional phrases together as a unit because the meaning of a phrase cannot be obtained straightforwardly from the constituent words. In this work we are investigate methods of determining semantic compositionality of verb-object [1] combinations on a continuum following previous research in this direction (McCarthy et al., 2003; Venkatapathy and Joshi, 2005).

Much previous research has used a combination of statistics and distributional approaches whereby distributional similarity is used to compare the constituents of the multiword with the multiword itself. In this paper, we will investigate the use of selectional preferences of verbs. We will use the preferences to find atypical verb-object combinations as we anticipate that such combinations are more likely to be non-compositional.

[1] We use object to refer to direct objects.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 369–379, Prague, June 2007. ©2007 Association for Computational Linguistics

Selectional preferences of predicates have been modelled using the man-made thesaurus WordNet (Fellbaum, 1998), see for example (Resnik, 1993; Li and Abe, 1998; Abney and Light, 1999; Clark and Weir, 2002). There are also distributional approaches which use co-occurrence data to cluster distributionally similar words together. The cluster output can then be used as classes for selectional preferences (Pereira et al., 1993), or one can directly use frequency information from distributionally similar words for smoothing (Grishman and Sterling, 1994).

We used three different types of probabilistic models, which vary in the classes selected for representation over which the probability distribution of the argument heads [2] is estimated. Two use WordNet and the other uses the entries in a thesaurus of distributionally similar words acquired automatically following (Lin, 1998). The first method is due to Li and Abe (1998). The classes over which the probability distribution is calculated are selected according to the minimum description length principle (MDL) which uses the argument head tokens for finding the best classes for representation. This method has previously been tried for modelling compositionality of verb-particle constructions (Bannard, 2002).

The other two methods (we refer to them as 'type-based') also calculate a probability distribution using argument head tokens but they select the classes over which the distribution is calculated using the number of argument head types (of a verb in a corpus) in a given class, rather than the number of argument head tokens in contrast to previous WordNet models (Resnik, 1993; Li and Abe, 1998; Clark and Weir, 2002). For example, if the object slot of the verb *park* contains the argument heads { *car, car, car, car, van, jeep* } then the type-based models use the word type "*car*" only once when determining the classes over which the probability distribution is to be estimated. Classes are selected which maximise the number of types that they cover, rather than the number of tokens. This is done to avoid the selectional preferences being heavily influenced by noise from highly frequent arguments which may be polysemous and some or all of their meanings may not be semantically related to the 'prototypical' arguments of the verb. For example *car* has a **gondola** sense in WordNet.

The third method uses entries in a distributional thesaurus rather than classes from WordNet. The entries used as classes for representation are selected by virtue of the number of argument types they encompass. As with the WordNet models, the tokens are used to estimate a probability distribution over these entries.

In the next section, we discuss related work on identifying compositionality. In section 3, we describe the methods we are using for acquiring our models of selectional preference. In section 4, we test our models on a dataset used in previous research. We compare the three types of models individually and also investigate the best performing model when used in combination with other features used in previous research. We conclude in section 5.

2 Related Work

Most previous work using distributional approaches to compositionality either contrasts distributional information of candidate phrases with constituent words (Schone and Jurafsky, 2001; Bannard et al., 2003; Baldwin et al., 2003; McCarthy et al., 2003) or uses distributionally similar words to detect non-productive phrases (Lin, 1999).

Lin (1999) used his method (Lin, 1998) for automatic thesaurus construction. He identified candidate phrases involving several open-class words output from his parser and filtered these by the log-likelihood statistic. Lin proposed that if there is a phrase obtained by substitution of either the head or modifier in the phrase with a 'nearest neighbour' from the thesaurus then the mutual information of this and the original phrase must be significantly different for the original phrase to be considered non-compositional. He evaluated the output manually.

As well as distributional similarity, researchers have used a variety of statistics as indicators of non-compositionality (Blaheta and Johnson, 2001; Krenn and Evert, 2001). Fazly and Stevenson (2006) use statistical measures of syntactic behaviour to gauge whether a verb and noun combination is likely to be a idiom. Although they are not specifically detecting compositionality, there is a strong corre-

[2] Argument heads are the nouns occurring in the object slot of the target verb.

lation between syntactic rigidity and semantic idiosyncrasy.

Venkatapathy and Joshi (2005) combine different statistical and distributional methods using support vector machines (SVMs) for identifying non-compositional verb-object combinations. They explored seven features as measures of compositionality:

1. frequency
2. pointwise mutual information (Church and Hanks, 1990),
3. least mutual information difference with similar collocations, based on (Lin, 1999) and using Lin's thesaurus (Lin, 1998) for obtaining the similar collocations.
4. The distributed frequency of an object, which takes an average of the frequency of occurrence with an object over all verbs occurring with the object above a threshold.
5. distributed frequency of an object, using the verb, which considers the similarity between the target verb and the verbs occurring with the target object above the specified threshold.
6. a latent semantic approach (LSA) based on (Schütze, 1998; Baldwin et al., 2003) and considering the dissimilarity of the verb-object pair with its constituent verb
7. the same LSA approach, but considering the similarity of the verb-object pair with the verbal form of the object (to capture support verb constructions e.g. *give a smile*

Venkatapathy and Joshi (2005) produced a dataset of verb-object pairs with human judgements of compositionality. We say more about this dataset and Venkatapathy and Joshi's results in section 4 since we use the dataset for our experiments.

In this paper, we investigate the use of selectional preferences to detect compositionality. Bannard (2002) did some pioneering work to try and establish a link between the compositionality of verb particle constructions and the selectional preferences of the multiword and its constituent verb. His results were hampered by models based on (Li and Abe, 1998) which involved rather uninformative models at the roots of WordNet. There are several reasons for this. The classes for the model are selected using MDL by compromising between a simple model with few classes and one which explains the data well. The models are particularly affected by the quantity of data available (Wagner, 2002). Also noise from frequent but idiosyncratic or polysemous arguments weakens the signal. There is scope for experimenting with other approaches such as (Clark and Weir, 2002), however, we feel a type-based approach is worthwhile to avoid the noise introduced from frequent but polysemous arguments and bias from highly frequent arguments which might be part of a multiword rather than a prototypical argument of the predicate in question, for example *eat hat*. In contrast to Bannard, our experiments are with verb-object combinations rather than verb particle constructions. We compare Li and Abe models with WordNet models which use the number of argument types to obtain the classes for representation of the selectional preferences. In addition to experiments with these WordNet models, we propose models using entries in distributional thesauruses for representing preferences.

3 Three Methods for Acquiring Selectional Preferences

All models were acquired from verb-object data extracted using the RASP parser (Briscoe and Carroll, 2002) from the 90 million words of written English from the BNC (Leech, 1992). We extracted verb and common noun tuples where the noun is the argument head of the object relation. The parser was also used to extract the grammatical relation data used for acquisition of the thesaurus described below in section 3.3.

3.1 TCMS

This approach is a reimplementation of Li and Abe (1998). Each selectional preference model (referred to as a tree cut model, or TCM) comprises a set of disjunctive noun classes selected from all the possibilities in the WordNet hyponym hierarchy [3] using MDL (Rissanen, 1978). The TCM covers all the

[3] We use WordNet version 2.1 for the work in this paper.

noun senses in the WordNet hierarchy and is associated with a probability distribution over these noun senses in the hierarchy reflecting the argument head data occurring in the given grammatical relationship with the specified verb. MDL finds the classes in the TCM by considering the cost measured in bits of describing both the model and the argument head data encoded in the model. A compromise is made by having as simple a model as possible using classes further up the hierarchy whilst also providing a good model for the set of argument head tokens (TK).

The classes are selected by recursing from the top of the WordNet hierarchy comparing the cost (or description length) of using the mother class to the cost of using the hyponym daughter classes. In any path, the mother is preferred unless using the daughters would reduce the cost. If using the daughters for the model is less costly than the mother then the recursion continues to compare the cost of the hyponyms beneath.

The cost (or description length) for a set of classes is calculated as the model description length (mdl) and the data description length (ddl) [4] :-

$$\begin{array}{cc} mdl & + \, ddl \\ \frac{k}{2} \times \log |TK| & + - \sum_{tk \in TK} \log p(tk) \end{array} \tag{1}$$

k, is the number of WordNet classes being currently considered for the TCM minus one. The MDL method uses the size of TK on the assumption that a larger dataset warrants a more detailed model. The cost of describing the argument head data is calculated using the log of the probability estimate from the classes currently being considered for the model. The probability estimate for a class being considered for the model is calculated using the cumulative frequency of all the hyponym nouns under that class that occur in TK, divided by the number of noun senses that these nouns have, to account for their polysemy. This cumulative frequency is also divided by the total number of noun hyponyms under that class in WordNet to obtain a smoothed estimate for all nouns under the class. The probability of the class is obtained by dividing this frequency estimate by the total frequency of the argument heads. The algorithm is described fully by Li and Abe (1998).

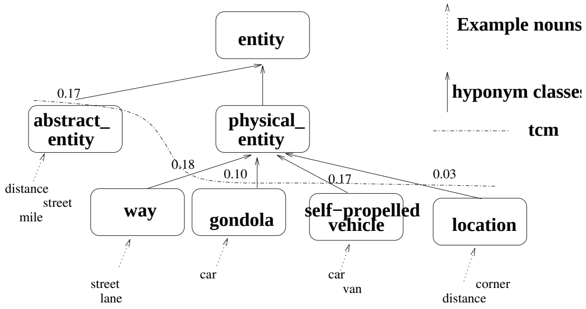


Figure 1: portion of the TCM for the objects of *park*.

A small portion of the TCM for the object slot of *park* is shown in figure 1. WordNet classes are displayed in boxes with a label which best reflects the meaning of the class. The probability estimates are shown for the classes on the TCM. Examples of the argument head data are displayed below the WordNet classes with dotted lines indicating membership at a hyponym class beneath these classes. We cannot show the full TCM due to lack of space, but we show some of the higher probability classes which cover some typical nouns that occur as objects of *park*. Note that probability under the classes **abstract_entity**, **way** and **location** arise because of a systematic parsing error where adverbials such as *distance* in *park illegally some distance from the railway station* are identified by the parser as objects. Systematic noise from the parser has an impact on all the selectional preference models described in this paper.

3.2 WNPROTOs

We propose a method of acquiring selectional preferences which instead of covering all the noun senses in WordNet, just gives a probability distribution over a portion of prototypical classes, we refer to these models as WNPROTOs. A WNPROTO consists of classes within the noun hierarchy which have the highest proportion of word types occurring in the argument head data, rather than using the number of tokens, or frequency, as is used for the TCMs. This allows less frequent, but potentially informative arguments to have some bearing on the models acquired to reduce the impact of highly frequent but polysemous arguments. We then used the frequency data to populate these selected classes.

[4] See (Li and Abe, 1998) for a full explanation.

The classes (C) in the WNPROTO are selected from those which include at least a threshold of 2 argument head types [5] occurring in the training data. Each argument head in the training data is disambiguated according to whichever of the WordNet classes it occurs at or under which has the highest 'type ratio'. Let TY be the set of argument head types in the object slot of the verb for which we are acquiring the preference model. The type ratio for a class (c) is the ratio of noun types ($ty \in TY$) occurring in the training data also listed at or beneath that class in WordNet to the total number of noun types listed at or beneath that particular class in WordNet ($wn_{ty} \in c$). The argument types attested in the training data are divided by the number of WordNet classes that the noun ($classes(ty)$) belongs to, to account for polysemy in the training data.

$$type\ ratio(c) = \frac{\sum_{ty \in TY \in c} \frac{1}{|classes(ty)|}}{|wn_{ty} \in c|} \qquad (2)$$

If more than one class has the same type ratio then the argument is not used for calculating the probability of the preference model. In this way, only arguments that can be disambiguated are used for calculating the probability distribution. The advantage of using the type ratio to determine the classes used to represent the model and to disambiguate the arguments is that it prevents high frequency verb noun combinations from masking the information from prototypical but low frequency arguments. We wish to use classes which are as representative of the argument head types as possible to help detect when an argument head is not related to these classes and is therefore more likely to be non-compositional.

For example, the class **motor_vehicle** is selected for the WNPROTO model of the object slot of *park* even though there are 5 meanings of *car* in WordNet including **elevator_car** and **gondola**. There are 174 occurrences of *car* which overwhelms the frequency of the other objects (e.g. *van* 11, *vehicle* 8) but by looking for classes with a high proportion of types (rather than word tokens) *car* is disambiguated appropriately and the class **motor_vehicle** is selected for representation.

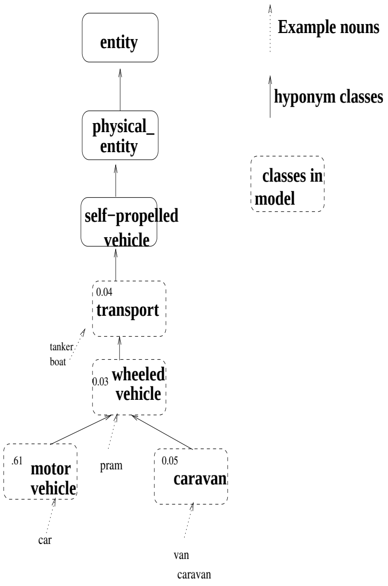


Figure 2: Part of WNPROTO for the object slot of *park*

The relative frequency of each class is obtained from the set of disambiguated argument head tokens and used to provide the probability distribution over this set of classes. Note that in WNPROTO, classes can be subsumed by others in the hyponym hierarchy. The probability assigned to a class is applicable to any descendants in the hyponym hierarchy, except those within any hyponym classes within the WNPROTO. The algorithm for selecting C and calculating the probability distribution is shown as Algorithm 1. Note that we use brackets for comments.

In figure 2 we show a small portion of the WNPROTO for *park*. Again, WordNet classes are displayed in boxes with a label which best reflects the meaning of the class. The probability estimates are shown in the boxes for all the classes included in the WNPROTO. The classes in the WNPROTO model are shown with dashed lines. Examples of the argument head data are displayed below the WordNet classes with dotted lines indicating membership at a hyponym class beneath these classes. We cannot show the full WNPROTO due to lack of space, but we show some of the classes with higher probability which cover some typical nouns that occur as objects of *park*.

[5] We have experimented with a threshold of 3 and obtained similar results.

Algorithm 1 WNPROTO algorithm

```
$C = ()$ {classes in WNPROTO}
$D = ()$ {disambiguated $ty \in TY$}
$fD = 0$ {frequency of disambiguated items}
$TY =$ argument head types {nouns occurring as objects of verb, with associated frequencies}
$C1 \in WordNet$
$where\ |ty \in TY occurring\ in\ c \in C1| > 1$
for all $ty \in TY$ do
    find $c \in classes(ty) \in C1\ where\ c = \text{argmax}_c\ typeratio(c)$
    if $c$ & $c \notin C$ then
        add $c$ to $C$
        add $ty \leftrightarrow c$ to $D$ {Disambiguated $ty$ with $c$}
    end if
end for
for all $c \in C$ do
    if $|ty \leftrightarrow c \in D| > 1$ then
        $fD = fD + frequency(ty)$ {sum frequencies of types under classes to be used in model}
    else
        remove $c$ from $C$ {classes with less than two disambiguated nouns are removed}
    end if
end for
for all $c \in C$ do
    $p(c) = \frac{frequency\text{-}of\text{-}all\text{-}tys\text{-}disambiguated\text{-}to\text{-}class(c,D)}{fD}$ {calculating class probabilities}
end for
```

Algorithm 2 DSPROTO algorithm

```
$C = ()$ {classes in DSPROTO}
$D = ()$ {disambiguated $ty \in TY$}
$fD = 0$ {frequency of disambiguated items}
$TY =$ argument head types {nouns occurring as objects of verb, with associated frequencies}
$C1$ = $cty \in TY\ where\ num\text{-}types\text{-}in\text{-}thesaurus(cty, TY) > 1$
order $C1$ by $num\text{-}types\text{-}in\text{-}thesaurus(cty, TY)$ {classes ordered by coverage of argument head types}
for all $cty \in ordered\, C1$ do
    $Dcty = ()$ {disambiguated for this class}
    for all $ty \in TY where\ in\text{-}thesaurus\text{-}entry(cty, ty)$ do
        if $ty \notin D$ then
            add $ty$ to $Dcty$ {types disambiguated to this class only if not disambiguated by a class used already}
        end if
    end for
    if $|Dcty| > 1$ then
        add $cty$ to $C$
        for all $ty \in Dcty$ do
            add $ty \leftrightarrow cty$ to $D$ {Disambiguated $ty$ with $cty$}
            $fD = fD + frequency(ty)$
        end for
    end if
end for
for all $cty \in C$ do
    $p(cty) = \frac{frequency\text{-}of\text{-}all\text{-}tys\text{-}disambiguated\text{-}to\text{-}class(cty,D)}{fD}$ {calculating class probabilities}
end for
```

3.3 DSPROTOs

We use a thesaurus acquired using the method proposed by Lin (1998). For input we used the grammatical relation data from automatic parses of the BNC. For each noun we considered the co-occurring verbs in the object and subject relation, the modifying nouns in noun-noun relations and the modifying adjectives in adjective-noun relations. Each thesaurus entry consists of the target noun and the 50 most similar nouns, according to Lin's measure of distributional similarity, to the target.

The argument head noun types (TY) are used to find the entries in the thesaurus as the 'classes' (C) of the selectional preference for a given verb. As with WNPROTOs, we only cover argument types which form coherent groups with other argument types since we wish i) to remove noise and ii) to be able to identify argument types which are not related with the other types and therefore may be non-compositional. As our starting point we only consider an argument type as a class for C if its entry in the thesaurus covers at least a threshold of 2 types. [6]

To select C we use a best first search. This method processes each argument type in TY in order of the number of the other argument types from TY that it has in its thesaurus entry of 50 similar nouns. An argument head is selected as a class for C ($cty \in C$) [7] if it covers at least 2 of the argument heads that are not in the thesaurus entries of any of the other classes already selected for C. Each argument head is disambiguated by whichever class in C under which it is listed in the thesaurus and which has the largest number of the TY in its thesaurus entry. When the algorithm finishes processing the ordered argument heads to select C, all argument head types are disambiguated by C apart from those which after disambiguation occur in isolation in a class without other argument types. Finally a probability distribution over C is estimated using the frequency (tokens) of argument types that occur in the thesaurus entries for any $cty \in C$. If an argument type occurs in the entry of more than one cty then it is assigned to whichever of these has the largest number of disambiguated argument head types and its token frequency is attributed to that class. We show the algorithm as Algorithm 2.

class ($p(c)$)	disambiguated objects (freq)
van (0.86)	car (174) van (11) vehicle (8) ...
mile (0.05)	street (5) distance (4) mile (1) ...
yard (0.03)	corner (4) lane (3) door (1)
backside (0.02)	backside (2) bum (1) butt (1) ...

Figure 3: First four classes of DSPROTO model for *park*

The algorithms for WNPROTO algorithm 1 and DSPROTO (algorithm 2) differ because of the nature of the inventories of candidate classes (WordNet and the distributional thesaurus). There are a great many candidate classes in WordNet. The WNPROTO algorithm selects the classes from all those that the argument heads belong to directly and indirectly by looping over all argument types to find the class that disambiguates each by having the largest type ratio calculated using the undisambiguated argument heads. The DSPROTO only selects classes from the fixed set of argument types. The algorithm loops over the argument types with at least two argument heads in the thesaurus entry and ordered by the number of undisambiguated argument heads in the thesaurus entry. This is a best first search to minimise the number of argument heads used in C but maximise the coverage of argument types.

In figure 3, we show part of a DSPROTO model for the object of *park*. [8] Note again that the class **mile** arises because of a systematic parsing error where adverbials such as *distance* in *park illegally some distance from the railway station* are identified by the parser as objects.

4 Experiments

Venkatapathy and Joshi (2005) produced a dataset of verb-object pairs with human judgements of compositionality. They obtained values of r_s between 0.111 and 0.300 by individually applying the 7 features described above in section 2. The best correlation was given by feature 7 and the second best was feature 3. They combined all 7 features using SVMs and splitting their data into test and training data and achieve a r_s of 0.448, which demonstrates

[6] As with the WNPROTOs, we experimented with a value of 3 for this threshold and obtained similar results.

[7] We use cty for the classes of the DSPROTO. These classes are simply groups of nouns which occur under the entry of a noun (ty) in the thesaurus.

[8] We cannot show the full model due to lack of space.

significantly better correlation with the human gold-standard than any of the features in isolation

We evaluated our selectional preference models using the verb-object pairs produced by Venkatapathy and Joshi (2005). [9] This dataset has 765 verb-object collocations which have been given a rating between 1 and 6, by two annotators (both fluent speakers of English). Kendall's Tau (Siegel and Castellan, 1988) was used to measure agreement, and a score of 0.61 was obtained which was highly significant. The ranks of the two annotators gave a Spearman's rank-correlation coefficient (r_s) of 0.71.

The Verb-Object pairs included some adjectives (e.g. *happy, difficult, popular*), pronouns and complements e.g. *become director*. We used the subset of 638 verb-object pairs that involved common nouns in the object relationship since our preference models focused on the object relation for common nouns. For each verb-object pair we used the preference models acquired from the RASP parses of the BNC to obtain the probability of the class that this object occurs under. Where the object noun is a member of several classes ($classes(noun) \in C$) in the model, the class with the largest probability is used. Note though that for WNPROTOs we have the added constraint that a hyponym class from C is selected in preference to a hypernym in C. Compositionality of an object noun and verb is computed as:-

$$comp(noun, verb) = max_{c \in classes(noun) \in C}\ p(c|verb) \quad (3)$$

We use the probability of the class, rather than an estimate of the probability of the object, because we want to determine how likely any word belonging to this class might occur with the given verb, rather than the probability of the specific noun which may be infrequent, yet typical, of the objects that occur with this verb. For example, *convertible* may be an infrequent object of *park*, but it is quite likely given its membership of the class **motor_vehicle**. We do not want to assume anything about the frequency of non-compositional verb-object combinations, just that they are unlikely to be members of classes which represent prototypical objects. We will contrast these models with a baseline frequency feature used by Venkatapathy and Joshi.

method	r_s	$p <$ (one tailed)
selectional preferences		
TCM	0.090	0.0119
WNPROTO	0.223	0.00003
DSPROTO	**0.398**	0.00003
features from V&J		
frequency (f1)	0.141	0.00023
MI (f2)	**0.274**	0.00003
Lin99 (f3)	0.139	0.00023
LSA2 (f7)	0.209	0.00003
combination with SVM		
f2,3,7	0.413	0.00003
f1,2,3,7	0.419	0.00003
DSPROTO f1,2,3,7	**0.454**	0.00003

Table 1: Correlation scores for 638 verb object pairs

We use our selectional preference models to provide the probability that a candidate is representative of the typical objects of the verb. That is, if the object might typically occur in such a relationship then this should lessen the chance that this verb-object combination is non-compositional. We used the probability of the classes from our 3 selectional preference models to rank the pairs and then used Spearman's rank-correlation coefficient (r_s) to compare these ranks with the ranks from the gold-standard.

Our results for the three types of preference models are shown in the first section of table 1. [10] All the correlation values are significant, but we note that using the type based selectional preference models achieves a far greater correlation than using the TCMs. The DSPROTO models achieve the best results which is very encouraging given that they only require raw data and an automatic parser to obtain the grammatical relations.

We applied 4 of the features used by Venkatapathy and Joshi (2005) [11] and described in section 2 to our subset of 638 items. These features were ob-

[9] This verb-object dataset is available from http://www.cis.upenn.edu/~sriramv/mywork.html.

[10] We show absolute values of correlation following (Venkatapathy and Joshi, 2005).

[11] The other 3 features performed less well on this dataset so we do not report the details here. This seems to be because they worked particularly well with the adjective and pronoun data in the full dataset.

tained using the same BNC dataset used by Venkatapathy and Joshi which was obtained using Bikel's parser (Bikel, 2004). We obtained correlation values for these features as shown in table 1 under V&J. These features are feature 1 frequency, feature 2 pointwise mutual information, feature 3 based on (Lin, 1999) and feature 7 LSA feature which considers the similarity of the verb-object pair with the verbal form of the object. Pointwise mutual information did surprisingly well on this 84% subset of the data, however the DSPROTO preferences still outperformed this feature. We combined the DSPROTO and V&J features with an SVM ranking function and used 10 fold cross validation as Venkatapathy and Joshi did. We contrast the result with the V&J features without the preference models. The results in the bottom section of table 1 demonstrate that the preference models can be combined with other features to produce optimal results.

5 Conclusions and Directions for Future Work

We have demonstrated that the selectional preferences of a verbal predicate can be used to indicate if a specific combination with an object is non-compositional. We have shown that selectional preference models which represent prototypical arguments and focus on argument types (rather than tokens) do well at the task. Models produced from distributional thesauruses are the most promising which is encouraging as the technique could be applied to a language without a man-made thesaurus. We find that the probability estimates from our models show a highly significant correlation, and are very promising for detecting non-compositional verb-object pairs, in comparison to individual features used previously.

Further comparison of WNPROTOs and DSPROTOs to other WordNet models are warranted to contrast the effect of our proposal for disambiguation using word types with iterative approaches, particularly those of Clark and Weir (2002). A benefit of the DSPROTOs is that they do not require a hand-crafted inventory. It would also be worthwhile comparing the use of raw data directly, both from the BNC and from google's Web 1T corpus (Brants and Franz, 2006) since web counts have been shown to outperform the Clark and Weir models on a pseudo-disambiguation task (Keller and Lapata, 2003).

We believe that preferences should NOT be used in isolation. Whilst a low preference for a noun may be indicative of peculiar semantics, this may not always be the case, for example *chew the fat*. Certainly it would be worth combining the preferences with other measures, such as syntactic fixedness (Fazly and Stevenson, 2006). We also believe it is worth targeting features to specific types of constructions, for example light verb constructions undoubtedly warrant special treatment (Stevenson et al., 2003)

The selectional preference models we have proposed here might also be applied to other tasks. We hope to use these models in tasks such as diathesis alternation detection (McCarthy, 2000; Tsang and Stevenson, 2004) and contrast with WordNet models previously used for this purpose.

6 Acknowledgements

We acknowledge support from the Royal Society UK for a Dorothy Hodgkin Fellowship to the first author. We thank the anonymous reviewers for their constructive comments on this work.

References

Steven Abney and Marc Light. 1999. Hiding a semantic class hierarchy in a Markov model. In *Proceedings of the ACL Workshop on Unsupervised Learning in Natural Language Processing*, pages 1–8.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL Workshop on multiword expressions: analysis, acquisition and treatment*, pages 89–96.

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In *Proceedings of the ACL Workshop on multiword expressions: analysis, acquisition and treatment*, pages 65–72.

Colin. Bannard. 2002. Statistical techniques for automatically inferring the semantics of verb-particle constructions. Technical Report WP-2002-06, University of Edinburgh, School of Informatics. http://lingo.stanford.edu/pubs/WP-2002-06.pdf.

Colin Bannard. 2005. Learning about the meaning of verb-particle constructions from corpora. *Computer Speech and Language*, 19(4):467–478.

Daniel M. Bikel. 2004. A distributional analysis of a lexicalized statistical parsing model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain, July. Association for Computational Linguistics.

Don Blaheta and Mark Johnson. 2001. Unsupervised learning of multi-word verbs. In *Proceedings of the ACL Workshop on Collocations*, pages 54–60, Toulouse, France.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram corpus version 1.1. Technical Report.

Edward Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1499–1504, Las Palmas, Canary Islands, Spain.

Kenneth Church and Patrick Hanks. 1990. Word association norms, mutual information and lexicography. *Computational Linguistics*, 19(2):263–312.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 337–344, Trento, Italy, April.

Christiane Fellbaum, editor. 1998. *WordNet, An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

Ralph Grishman and John Sterling. 1994. Generalizing automatically generated selectional patterns. In *Proceedings of the 15th International Conference of Computational Linguistics. COLING-94*, volume I, pages 742–747.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Brigitte Krenn and Stefan Evert. 2001. Can we do better than frequency? A case study on extracting PP-verb collocations. In *Proceedings of the ACL Workshop on Collocations*, pages 39–46, Toulouse, France.

Geoffrey Leech. 1992. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13.

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics*, 24(2):217–244.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL 98*, Montreal, Canada.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of ACL-99*, pages 317–324, Univeristy of Maryland, College Park, Maryland.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 03 Workshop: Multiword expressions: analysis, acquisition and treatment*, pages 73–80.

Diana McCarthy. 2000. Using semantic preferences to identify verbal participation in role switching alternations. In *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics. (NAACL)*, pages 256–263, Seattle,WA.

Fernando Pereira, Nattali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Philip Resnik. 1993. *Selection and Information: A Class-Based Approach to Lexical Relationships*. Ph.D. thesis, University of Pennsylvania.

Jorma Rissanen. 1978. Modelling by shortest data description. *Automatica*, 14:465–471.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2002)*, pages 1–15, Mexico City, Mexico.

Patrick Schone and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 100–108, Hong Kong.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Sidney Siegel and N. John Castellan. 1988. *Non-Parametric Statistics for the Behavioral Sciences*. McGraw-Hill, New York.

Suzanne Stevenson, Afsaneh Fazly, and Ryan North. 2003. Statistical measures of the semi-productivity of light verb constructions. In *Proceedings of the ACL 2004 Workshop on Multiword Expressions: Integrating Processing*, Barcelona, Spain.

Vivian Tsang and Suzanne Stevenson. 2004. Using selectional profile distance to detect verb alternations. In *Proceedings of NAACL Workshop on Computational Lexical Semantics (CLS-04)*, pages 30–37, Boston, MA.

Sriram Venkatapathy and Aravind K. Joshi. 2005. Measuring the relative compositionality of verb-noun (v-n) collocations by integrating features. In *Proceedings of the joint conference on Human Language Technology and Empirical methods in Natural Language Processing*, pages 899–906, Vancouver, B.C., Canada.

Andreas Wagner. 2002. Learning thematic role relations for wordnets. In *Proceedings of ESSLLI-2002 Workshop on Machine Learning Approaches in Computational Linguistics*, Trento.

Explorations in Automatic Book Summarization

Rada Mihalcea and **Hakan Ceylan**
Department of Computer Science
University of North Texas
rada@cs.unt.edu, hakan@unt.edu

Abstract

Most of the text summarization research carried out to date has been concerned with the summarization of short documents (e.g., news stories, technical reports), and very little work if any has been done on the summarization of very long documents. In this paper, we try to address this gap and explore the problem of book summarization. We introduce a new data set specifically designed for the evaluation of systems for book summarization, and describe summarization techniques that explicitly account for the length of the documents.

1 Introduction

Books represent one of the oldest forms of written communication and have been used since thousands of years ago as a means to store and transmit information. Despite this fact, given that a large fraction of the electronic documents available online and elsewhere consist of short texts such as Web pages, news articles, scientific reports, and others, the focus of natural language processing techniques to date has been on the automation of methods targeting short documents. We are witnessing however a change: an increasingly larger number of books become available in electronic format, in projects such as Gutenberg (http://www.gutenberg.org), Google Book Search (http://books.google.com), or the Million Books project (http://www.archive.org/details/millionbooks). Similarly, a large number of the books published in recent years are often available – for purchase or through libraries – in electronic format. This means that the need for language processing techniques able to handle very large documents such as books is becoming increasingly important.

In this paper, we address the problem of *book summarization*. While there is a significant body of research that has been carried out on the task of text summarization, most of this work has been concerned with the summarization of *short* documents, with a particular focus on news stories. However, books are different in both length and genre, and consequently different summarization techniques are required. In fact, the straight-forward application of a current state-of-the-art summarization tool leads to poor results – a mere 0.348 F-measure compared to the baseline of 0.325 (see the following sections for details). This is not surprising since these systems were developed specifically for the summarization of short news documents.

The paper makes two contributions. First, we introduce a new data set specifically designed for the evaluation of book summaries. We describe the characteristics of a new benchmark consisting of books with manually constructed summaries, and we calculate and provide lower and upper performance bounds on this data set. Second, after briefly describing a summarization system that has been successfully used for the summarization of short documents, we show how techniques that take into account the length of the documents can be used to significantly improve the performance of this system.

2 Related Work

Automatic summarization has received a lot of attention from the natural language processing commu-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 380–389, Prague, June 2007. ©2007 Association for Computational Linguistics

nity, ever since the early approaches to automatic abstraction that laid the foundations of the current text summarization techniques (Luhn, 1958; Edmunson, 1969). The literature typically distinguishes between *extraction*, concerned with the identification of the information that is important in the input text; and *abstraction*, which involves a generation step to add fluency to a previously compressed text (Hovy and Lin, 1997). Most of the efforts to date have been concentrated on the extraction step, which is perhaps the most critical component of a successful summarization algorithm, and this is the focus of our current work as well.

To our knowledge, no research work to date was specifically concerned with the automatic summarization of *books*. There is, however, a large and growing body of work concerned with the summarization of short documents, with evaluations typically focusing on news articles. In particular, a significant number of summarization systems have been proposed during the recent Document Understanding Conference exercises (DUC) – annual evaluations that usually draw the participation of 20–30 teams every year.

There are two main trends that can be identified in the summarization literature: *supervised* systems, that rely on machine learning algorithms trained on pre-existing document-summary pairs, and *unsupervised* techniques, based on properties and heuristics derived from the text.

Among the unsupervised techniques, typical summarization methods account for both the weight of the words in sentences, as well as the sentence position inside a document. These techniques have been successfully implemented in the centroid approach (Radev et al., 2004), which extends the idea of *tf.idf* weighting (Salton and Buckley, 1997) by introducing word centroids, as well as integrating other features such as position, first-sentence overlap and sentence length. More recently, graph-based methods that rely on sentence connectivity have also been found successful, using algorithms such as node degree (Salton et al., 1997) or eigenvector centrality (Mihalcea and Tarau, 2004; Erkan and Radev, 2004; Wolf and Gibson, 2004).

In addition to unsupervised methods, supervised machine learning techniques have also been used with considerable success. Assuming the availability of a collection of documents and their corresponding manually constructed summaries, these methods attempt to identify the key properties of a good summary, such as the presence of named entities, positional scores, or the location of key phrases. Such supervised techniques have been successfully used in the systems proposed by e.g. (Teufel and Moens, 1997; Hirao et al., 2002; Zhou and Hovy, 2003; D'Avanzo and Magnini, 2005).

In addition to short news documents, which have been the focus of most of the summarization systems proposed to date, work has been also carried out on the summarization of other types of documents. This includes systems addressing the summarization of e-mail threads (Wan and McKeown, 2004), online discussions (Zhou and Hovy, 2005), spoken dialogue (Galley, 2006), product reviews (Hu and Liu, 2004), movie reviews (Zhuang et al., 2006), or short literary fiction stories (Kazantseva and Szpakowicz, 2006). As mentioned before, we are not aware of any work addressing the task of automatic book summarization.

3 A Data Set for the Evaluation of Book Summarization

A first challenge we encountered when we started working on the task of book summarization was the lack of a suitable data set, designed specifically for the evaluation of summaries of long documents. Unlike the summarization of short documents, which benefits from the data sets made available through the annual DUC evaluations, we are not aware of any publicly available data sets that can be used for the evaluation of methods for book summarization.

The lack of such data sets is perhaps not surprising since even for humans the summarization of books is more difficult and time consuming than the summarization of short news documents. Moreover, books are often available in printed format and are typically protected by copyright laws that do not allow their reproduction in electronic format, which consequently prohibits their public distribution.

We constructed a data set starting from the observation that several English and literature courses make use of books that are sometimes also available in the form of abstracts – meant to ease the access of students to the content of the books. In particular, we have identified two main publishers that make summaries available online for books studied in the U.S. high-school and college systems: Grade Saver (http://www.gradesaver.com) and Cliff's Notes (http://www.cliffsnotes.com/). Fortunately, many of these books are classics that are already in the public domain, and thus for most of them we were able to find the online electronic version of the books on sites such as Gutenberg or Online Literature (http://www.online-literature.com).

For instance, the following is an example drawn from Cliff's Notes summary of *Bleak House* by Charles Dickens.

> On a raw November afternoon, London is enshrouded in heavy fog made harsher by chimney smoke. The fog seems thickest in the vicinity of the High Court of Chancery. The court, now in session, is hearing an aspect of the case of Jarndyce and Jarndyce. A ”little mad old woman” is, as always, one of the spectators. Two ruined men, one a ”sallow prisoner,” the other a man from Shropshire, appear before the court – to no avail. Toward the end of the sitting, the Lord High Chancellor announces that in the morning he will meet with ”the two young people” and decide about making them wards of their cousin....

Starting with the set of books that had a summary available from Cliff's Notes, we removed all the books that did not have an online version, and further eliminated those that did not have a summary available from Grade Saver. This left us with a "gold standard" data set of 50 books, each of them with two manually created summaries.

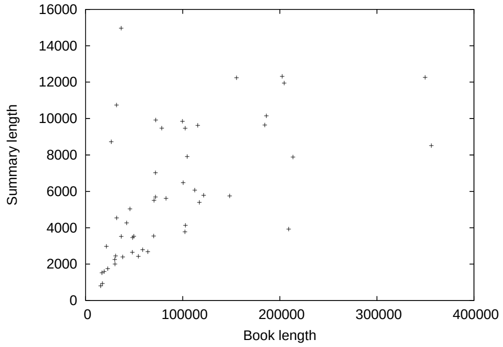


Figure 1: Summary and book lengths for 50 books

The books in this collection have an average length of 92,000 words, with summaries with an average length of 6,500 words (Cliff's Notes) and 7,500 words (Grade Saver). Figure 1 plots the length of the summaries (averaged over the two manual summaries) with respect to the length of the books. As seen in the plot, most of the books have a length of 50,000-150,000 words, with a summary of 2,000–6,000 words, corresponding to a compression rate of about 5-15%. There are also a few very long books, with more than 150,000 words, for which the summaries tend to become correspondingly longer.

3.1 Evaluation Metrics

For the evaluation, we use the ROUGE evaluation toolkit. ROUGE is a method based on Ngram statistics, found to be highly correlated with human evaluations (Lin and Hovy, 2003).[1] Throughout the paper, the evaluations are reported using the ROUGE-1 setting, which seeks unigram matches between the generated and the reference summaries, and which was found to have high correlation with human judgments at a 95% confidence level. Additionally, the final system is also evaluated using the ROUGE-2 (bigram matches) and ROUGE-SU4 (non-contiguous bigrams) settings, which have been frequently used in the DUC evaluations.

In most of the previous summarization evaluations, the data sets were constructed specifically for the purpose of enabling system evaluations, and thus the length of the reference and the generated summaries was established prior to building the data set and prior to the evaluations. For instance, some of the previous DUC evaluations provided reference summaries of 100-word each, and required the participating systems to generate summaries of the same length.

However, in our case we have to deal with pre-existing summaries, with large summary-length variations across the 50 books and across the two reference summaries. To address this problem, we decided to keep one manual summary as the main reference (Grade Saver), and use the other summary (Cliff's Notes) as a way to decide on the length of the generated summaries. This means that for a given book, the Cliff's Notes summary and all the

[1] ROUGE is available at http://haydn.isi.edu/ROUGE/

automatically generated summaries have the same length, and they are all evaluated against the (possibly with a different length) Grade Saver summary. This way, we can also calculate an upper bound by comparing the two manual summaries against each other, and at the same time ensure a fair comparison between the automatically generated summaries and this upper bound.[2]

3.2 Lower and Upper Bounds

To determine the difficulty of the task on the 50 book data set, we calculate and report lower and upper bounds. The lower bound is determined by using a baseline summary constructed by including the first sentences in the book (also known in the literature as the *lead* baseline).[3] As mentioned in the previous section, all the generated summaries – including this baseline – have a length equal to the Cliff's Notes manual summary. The upper bound is calculated by evaluating Cliff's Notes manual summary against the reference Grade Saver summary. Table 1 shows the precision (P), recall (R), and F-measure (F) for these lower and upper bounds, calculated as average across the 50 books.

	P	R	F
Lower bound (lead baseline)	0.380	0.284	0.325
Upper bound (manual summary)	0.569	0.493	0.528

Table 1: Lower and upper bounds for the book summarization task, calculated on the 50 book data set

An automatic system evaluated on this data set is therefore expected to have an F-measure higher than the lower bound of 0.325, and it is unlikely to exceed the upper bound of 0.528 obtained with a human-generated summary.

4 An Initial Summarization System

Our first book summarization experiment was done using a re-implementation of an existing state-of-the-art summarization system. We decided to use the centroid-based method implemented in the MEAD system (Radev et al., 2004), for three main reasons. First, MEAD was shown to lead to good performance in several DUC evaluations, e.g., (Radev et al., 2003; Li et al., 2005). Second, it is an unsupervised method which, unlike supervised approaches, does not require training data (not available in our case). Finally, the centroid-based techniques implemented in MEAD can be optimized and made very efficient, which is an important aspect in the summarization of very long documents such as books.

The latest version of MEAD[4] uses features, classifiers and re-rankers to determine the sentences to include in the summary. The default features are centroid, position and sentence length. The centroid value of a sentence is the sum of the centroid values of the words in the sentence. The centroid value of a word is calculated by multiplying the term frequency (*tf*) of a word by the word's inverse document frequency (*idf*) obtained from the Topic Detection and Tracking (TDT) corpus. The *tf* of a word is calculated by dividing the frequency of a word in a document cluster by the number of documents in the cluster. The positional value P_i of a sentence is calculated using the formula (Radev et al., 2004):

$$P_i = \frac{n - i + 1}{n} * C_{max} \qquad (1)$$

where n represents the number of sentences in the document, i represents the position of the sentence inside the text, and C_{max} is the score of the sentence that has the maximum centroid value.

The summarizer combines these features to give a score to each sentence. The default setting consists of a linear combination of features that assigns equal weights to the centroid and the positional values, and only scores sentences that have more than nine words. After the sentences are scored, the re-rankers are used to modify the scores of a sentence depending on its relation with other sentences. The default re-ranker implemented in MEAD first ranks the sentences by their scores in descending order and iteratively adds the top ranked sentence if the sentence is not *too similar* to the already added sentences. This similarity is computed as a cosine similarity and by default the sentences that exhibit a cosine similarity higher than 0.7 are not added to the

[2] An alternative solution would be to determine the length of the generated summaries using a predefined compression rate (e.g., 10%). However, this again implies great variations across the lengths of the generated versus the manual summaries, which can result in large and difficult to interpret variations across the ROUGE scores.

[3] A second baseline that accounts for text segments is also calculated and reported in section 6.

[4] MEAD 3.11, http://www.summarization.com/mead/

summary. Note that although the MEAD distribution also includes an optional feature calculated using the LexRank graph-based algorithm (Erkan and Radev, 2004), this feature could not be used since it takes days to compute for very long documents such as ours, and thus its application was not tractable.

Although the MEAD system is publicly available for download, in order to be able to make continuous modifications easily and efficiently to the system as we develop new methods, we decided to write our own implementation. Our implementation differs from the original one in certain aspects. First, we determine document frequency counts using the British National Corpus (BNC) rather than the TDT corpus. Second, we normalize the sentence scores by dividing the score of a sentence by the length of the sentence, and instead we eliminate the sentence length feature used by MEAD. Note also that we do not take stop words into account when calculating the length of a sentence. Finally, since we are not doing *multi*-document summarization, we do not use a re-ranker in our implementation.

	P	R	F
MEAD (original download)	0.423	0.296	0.348
MEAD (our implementation)	0.435	0.323	0.369

Table 2: Summarization results using the MEAD system

Table 2 shows the results obtained on the 50 book data set using the original MEAD implementation, as well as our implementation. Although the performance of this system is clearly better than the baseline (see Table 1), it is nonetheless far below the upper bound. In the following section, we explore techniques for improving the quality of the generated summaries by accounting for the length of the documents.

5 Techniques for Book Summarization

We decided to make several changes to our initial system, in order to account for the specifics of the data set we work with. In particular, our data set consists of *very large* documents, and correspondingly the summarization of such documents requires techniques that account for their length.

5.1 Sentence Position In Very Large Documents

The general belief in the text summarization literature (Edmunson, 1969; Mani, 2001) is that the position of sentences in a text represents one of the most important sources of information for a summarization system. In fact, a summary constructed using the lead sentences was often found to be a competitive baseline, with only few systems exceeding this baseline during the recent DUC summarization evaluations.

Although the position of sentences in a document seems like a pertinent heuristic for the summarization of short documents, and in particular for the newswire genre as used in the DUC evaluations, our hypothesis is that this heuristic may not hold for the summarization of very long documents such as books. The style and topic may change several times throughout a book, and thus the leading sentences will not necessarily overlap with the essence of the document.

To test this hypothesis, we modified our initial system so that it does not account for the position of the sentences inside a document, but it only accounts for the weight of the constituent words. Correspondingly, the score of a sentence is determined only as a function of the word centroids, and excludes the positional score. Table 3 shows the average ROUGE scores obtained using the summarization system with and without the position scores.

	P	R	F
With positional scores	0.435	0.323	0.369
Without positional scores	0.459	0.329	0.383

Table 3: Summarization results with and without positional scores

As suspected, removing the position scores leads to a better overall performance, with an increase observed in both the precision and the recall of the system. Although the position in a document is a heuristic that helps the summarization of news stories and other short documents, it appears that the sentences located toward the beginning of a book are not necessarily useful for building the summary of a book.

5.2 Text Segmentation

A major difference between short and long documents stands in the frequent topic shifts typically observed in the later. While short stories are usually concerned with one topic at a time, long documents such as books often cover more than one topic. Thus, the intuition is that a summary should include content covering the important aspects of *all* the topics in the document, as opposed to only generic aspects relevant to the document as a whole. A system for the summarization of *long* documents should therefore extract key concepts from all the topics in the document, and this task is better performed when the topic boundaries are known prior to the summarization step.

To accomplish this, we augment our system with a text segmentation module that attempts to determine the topic shifts, and correspondingly splits the document into smaller segments. Note that although chapter boundaries are available in some of the books in our data set, this is not always the case as there are also books for which the chapters are not explicitly identified. To ensure an uniform treatment of the entire data set, we decided not to use chapter boundaries, and instead apply an automatic text segmentation algorithm.

While several text segmentation systems have been proposed to date, we decided to use a graph-based segmentation algorithm using normalized-cuts (Malioutov and Barzilay, 2006), shown to exceed the performance of alternative segmentation methods. Briefly, the segmentation algorithm starts by modeling the text as a graph, where sentences are represented as nodes in the graph, and inter-sentential similarities are used to draw weighted edges. The similarity between sentences is calculated using cosine similarity, with a smoothing factor that adds the counts of the words in the neighbor sentences. Words are weighted using an adaptation of the *tf.idf* metric, where a document is uniformly split into chunks that are used for the *tf.idf* computation. There are two parameters that have to be set in this algorithm: (1) the length in words of the blocks approximating sentences; and (2) the cut-off value for drawing edges between nodes. Since the method was originally developed for spoken lecture segmentation, we were not able to use the same parameters as suggested in (Malioutov and Barzilay, 2006). Instead, we used a development set of three books, and determined the optimal sentence word-length as 20 and the optimal cut-off value as 25, and these are the values used throughout our experiments.

Once the text is divided into segments, we generate a separate summary for each segment, and consequently create a final summary by collecting sentences from the individual segment summaries in a round-robin fashion. That is, starting with the ranked list of sentences generated by the summarization algorithm for each segment, we pick one sentence at a time from each segment summary until we reach the desired book-summary length.

A useful property of the normalized-cut segmentation algorithm is that one can decide apriori the number of segments to be generated, and so we can evaluate the summarization algorithm for different segmentation granularities. Figure 2 shows the average ROUGE-1 F-measure score obtained for summaries generated using one to 50 segments.

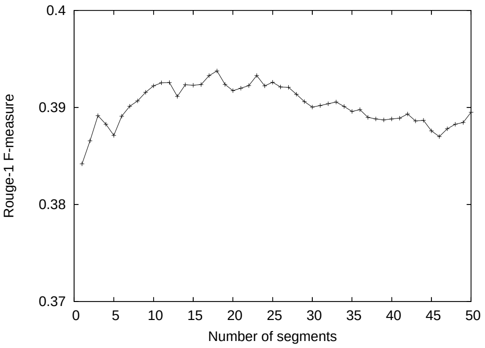


Figure 2: Summarization results for different segmentation granularities.

As seen in the figure, segmenting the text helps the summarization process. The average ROUGE-1 F-measure score raises to more than 0.39 F-measure for increasingly larger number of segments, with a plateau reached at approximately 15–25 segments, followed by a decrease when more than 30 segments are used.

In all the following evaluations, we segment each book into a constant number of 15 segments; in future work, we plan to consider more sophisticated methods for finding the optimal number of segments individually for each book.

5.3 Modified Term Weighting

An interesting characteristic of documents with topic shifts is that words do not have an uniform distribution across the entire document. Instead, their distribution can vary with the topic, and thus the weight of the words should change accordingly.

To account for the distribution of the words inside the entire book, as well as inside the individual topics (segments), we devised a weighting scheme that accounts for four factors: the *segment term frequency (stf)*, calculated as the number of occurrences of a word inside a segment; the *book term frequency (tf)*, determined as the number of occurrences of a word inside a book; the *inverse segment frequency (isf)*, measured as the inverse of the number of segments containing the word; and finally, the *inverse document frequency (idf)*, which takes into account the distribution of a word in a large external corpus (as before, we use the BNC corpus). A word weight is consequently determined by multiplying the book term frequency with the segment term frequency, and the result is then multiplied with the inverse segment frequency and the inverse document frequency. We refer to this weighting scheme as *tf.stf.idf.isf*.

Using this weighting scheme, we prevent a word from having the same score across the entire book, and instead we give a higher weight to its occurrences in segments where the word has a high frequency. For instance, the word *doctor* occurs 30 times in one of the books in our data set, which leads to a constant *tf.idf* score of 36.76 across the entire book. Observing that from these 30 occurrences, 19 appear in just one segment, the *tf.stf.idf.isf* weighting scheme will lead to a weight of 698.49 for that segment, much higher than e.g. the weight of 36 calculated for other segments that have only a few occurrences of this word.

	P	R	F
tf.idf weighting	0.463	0.339	0.391
tf.stf.idf.isf weighting	0.464	0.349	0.398

Table 4: Summarization results using a weighting scheme accounting for the distribution of words inside and across segments

Table 4 shows the summarization results obtained for the new weighting scheme (recall that all the results are calculated for a text segmentation into 15 segments).

5.4 Combining Summarization Methods

The next improvement we made was to bring an additional source of knowledge into the system, by combining the summarization provided by our current system with the summarization obtained from a different method.

We implemented a variation of a centrality graph-based algorithm for unsupervised summarization, which was successfully used in the past for the summarization of short documents. Very briefly, the TextRank system (Mihalcea and Tarau, 2004) – similar in spirit with the concurrently proposed LexRank method (Erkan and Radev, 2004) – works by building a graph representation of the text, where sentences are represented as nodes, and weighted edges are drawn using inter-sentential word overlap. An eigenvector centrality algorithm is then applied on the graph (e.g., PageRank), leading to a ranking over the sentences in the document. An impediment we encountered was the size of the graphs, which become intractably large and dense for very large documents such as books. In our implementation we decided to use a cut-off value for drawing edges between nodes, and consequently removed all the edges between nodes that are farther apart than a given threshold. We use a threshold value of 75, found to work best using the same development set of three books used before.

	P	R	F
Our system	0.464	0.349	0.398
TextRank	0.449	0.356	0.397
COMBINED	0.464	0.363	0.407

Table 5: Summarization results for individual and combined summarization algorithms

Using the same segmentation as before (15 segments), the TextRank method by itself did not lead to improvements over our current centroid-based system. Instead, since we noticed that the summaries generated with our system and with TextRank covered different sentences, we implemented a method that combines the top ranked sentences from the two methods. Specifically, the combination method picks one sentence at a time from the summary generated by our system for each segment, followed by one sentence selected from the summary generated by the TextRank method, and so on. The combination method also specifically avoids redundancy. Table 5 shows the results obtained with our current centroid-based system the TextRank method, as well as the combined method.

5.5 Segment Ranking

In the current system, all the segments identified in a book have equal weight. However, this might not always be the case, as there are sometimes topics inside the book that have higher importance, and which consequently should be more heavily represented in the generated summaries.

To account for this intuition, we implemented a segment ranking method that assigns to each segment a score reflecting its importance inside the book. The ranking is performed with a method similar to TextRank, using a random-walk model over a graph representing segments and segment similarities. The resulting segment scores are multiplied with the sentence scores obtained from the combined method described before, normalized over each segment, resulting in a new set of scores. The top ranked sentences over the entire book are then selected for inclusion in the summary. Table 6 shows the results obtained by using segment ranking.

	P	R	F
COMBINED	0.464	0.363	0.407
COMBINED + Segment Ranking	0.472	0.366	0.412

Table 6: Summarization results using segment ranking

6 Discussion

In addition to the ROUGE-1 metric, the quality of the summaries generated with our final summarization system was also evaluated using the ROUGE-2 and the ROUGE-SU4 metrics, which are frequently used in the DUC evaluations. Table 7 shows the figures obtained with ROUGE-1, ROUGE-2 and ROUGE-SU4 for our final system, for the original MEAD download, as well as for the lower and upper bounds. The table also shows an additional baseline determined by selecting the first sentences in each segment, using the segmentation into 15 segments as determined before. As it can be seen from the F-measure confidence intervals also shown in the table, the improvements obtained by our system with respect to both baselines and with respect to the MEAD system are statistically significant (as the confidence intervals do not overlap).

	P	R	F
	ROUGE-1		
Lower bound	0.380	0.284	0.325 [0.306,0.343]
Segment baseline	0.402	0.301	0.344 [0.328,0.366]
MEAD	0.423	0.296	0.348 [0.329,0.368]
Our system	**0.472**	**0.366**	**0.412 [0.394,0.428]**
Upper bound	0.569	0.493	0.528 [0.507,0.548]
	ROUGE-2		
Lower bound	0.035	0.027	0.031 [0.027,0.035]
Segment baseline	0.040	0.031	0.035 [0.031,0.038]
MEAD	0.039	0.029	0.033 [0.028,0.037]
Our system	**0.069**	**0.054**	**0.061 [0.055,0.067]**
Upper bound	0.112	0.097	0.104 [0.096,0.111]
	ROUGE-SU4		
Lower bound	0.096	0.073	0.083 [0.076,0.090]
Segment baseline	0.102	0.079	0.089 [0.082,0.093]
MEAD	0.106	0.076	0.088 [0.081,0.095]
Our system	**0.148**	**0.115**	**0.129 [0.121,0.138]**
Upper bound	0.210	0.182	0.195 [0.183,0.206]

Table 7: Evaluation of our final book summarization system using different ROUGE metrics. The table also shows: the lower bound (first sentences in the book); the segment baseline (first sentences in each segment); MEAD (original system download); the upper bound (manual summary). Confidence intervals for F-measure are also included.

Additionally, to determine the robustness of the results with respect to the number of reference summaries, we ran a separate evaluation where both the Grade Saver and the Cliff's Notes summaries were used as reference. As before, the length of the generated summaries was determined based on the Cliff's Notes summary. The F-measure figures obtained in this case using our summarization system were 0.402, 0.057 and 0.127 using ROUGE-1, ROUGE-2 and ROUGE-SU4 respectively. The F-measure figures calculated for the baseline using the first sentences in each segment were 0.340, 0.033 and 0.085. These figures are very close to those listed in Table 7 where only one summary was used as a reference, suggesting that the use of more than one reference summary does not influence the results.

Regardless of the evaluation metric used, the performance of our book summarization system is significantly higher than the one of an existing summarization system that has been designed for the summarization of short documents (MEAD). In fact, if we account for the upper bound of 0.528, the relative error rate reduction for the ROUGE-1 F-measure score obtained by our system with respect to MEAD is a significant 34.44%.

The performance of our system is mainly due to features that account for the length of the document: exclusion of positional scores, text segmentation and segment ranking, and a segment-based weighting scheme. An additional improvement is obtained by combining two different summarization methods. It is also worth noting that our system is efficient, taking about 200 seconds to apply the segmentation algorithm, plus an additional 65 seconds to generate the summary of one book.[5]

To assess the usefulness of our system with respect to the length of the documents, we analyzed the individual results obtained for books of different sizes. Averaging the results obtained for the shorter books in our collection, i.e., 17 books with a length between 20,000 and 50,000 words, the lead baseline gives a ROUGE-1 F-measure score of 0.337, our system leads to 0.378, and the upper bound is measured at 0.498, indicating a relative error rate reduction of 25.46% obtained by our system with respect to the lead baseline (accounting for the maximum achievable score given by the upper bound). Instead, when we consider only the books with a length over 100,000 words (16 books in our data set fall under this category), the lead baseline is determined as 0.347, our system leads to 0.418, and the upper bound is calculated as 0.552, which results in a higher 34.64% relative error rate reduction. This suggests that our system is even more effective for longer books, due perhaps to the features that specifically take into account the length of the books.

There are also cases where our system does not improve over the baseline. For instance, for the summarization of *Candide* by François Voltaire, our system achieves a ROUGE-1 F-measure of 0.361, which is slightly worse than the lead baseline of 0.368. In other cases however, the performance of our system comes close to the upper bound, as it is the case with the summarization of *The House of the Seven Gables* by Nathaniel Hawthorne, which has a lead baseline of 0.296, an upper bound of 0.457, and our system obtains 0.404. This indicates that a possible avenue for future research is to account for the characteristics of a book, and devise summarization methods that can adapt to the specifics of a given book such as length, genre, and others.

7 Conclusions

Although there is a significant body of work that has been carried out on the task of text summarization, most of the research to date has been concerned with the summarization of *short* documents. In this paper, we tried to address this gap and tackled the problem of book summarization.

We believe this paper made two important contributions. First, it introduced a new summarization benchmark, specifically targeting the evaluation of systems for book summarization.[6] Second, it showed that systems developed for the summarization of short documents do not fare well when applied to very long documents such as books, and instead a better performance can be achieved with a system that accounts for the length of the documents. In particular, the book summarization system we developed was found to lead to more than 30% relative error rate reduction with respect to an existing state-of-the-art summarization tool.

Given the increasingly large number of books available in electronic format, and correspondingly the growing need for tools for book summarization, we believe that the topic of automatic book summarization will become increasingly important. We hope that this paper will encourage and facilitate the development of an active line of research concerned with book summarization.

Acknowledgments

The authors are grateful to the Language and Information Technologies research group at the University of North Texas for useful discussions and feedback, and in particular to Carmen Banea for suggesting Cliff's Notes as a source of book summaries. This work was supported in part by a research grant from Google Inc. and by a grant from the Texas Advanced Research Program (#003594).

[5] Running times measured on a Pentium IV 3GHz, 2GB RAM.

[6] The data set is publicly available and can be downloaded from http://lit.csci.unt.edu/index.php/Downloads

References

E. D'Avanzo and B. Magnini. 2005. A keyphrase-based approach to summarization: The Lake system at DUC 2005. In *Proceedings of the Document Understanding Conference (DUC 2005)*.

H.P. Edmunson. 1969. New methods in automatic extracting. *Journal of the ACM*, 16(2):264–285.

G. Erkan and D. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain, July.

M. Galley. 2006. Automatic summarization of conversational multi-party speech. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006), AAAI/SIGART Doctoral Consortium*, Boston.

T. Hirao, Y. Sasaki, H. Isozaki, and E. Maeda. 2002. NTT's text summarization system for DUC-2002. In *Proceedings of the Document Understanding Conference 2002 (DUC 2002)*.

E. Hovy and C. Lin, 1997. *Automated text summarization in SUMMARIST.* Cambridge Univ. Press.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, Seattle, Washington.

A. Kazantseva and S. Szpakowicz. 2006. Challenges in evaluating summaries of short stories. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, Sydney, Australia.

W. Li, W. Li, B. Li, Q. Chen, and M. Wu. 2005. The Hong Kong Polytechnic University at DUC 2005. In *Proceedings of the Document Understanding Conference (DUC 2005)*, Vancouver, Canada.

C.Y. Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May.

H. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

I. Malioutov and R. Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 9–16.

I. Mani. 2001. *Automatic Summarization*. John Benjamins.

R. Mihalcea and P. Tarau. 2004. TextRank – bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain.

D. Radev, J. Otterbacher, H. Qi, and D. Tam. 2003. MEAD ReDUCs: Michigan at DUC 2003. In *Proceedings of the Document Understanding Conference (DUC 2003)*.

D. Radev, H. Jing, M. Stys, and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40.

G. Salton and C. Buckley. 1997. Term weighting approaches in automatic text retrieval. In *Readings in Information Retrieval*. Morgan Kaufmann Publishers, San Francisco, CA.

G. Salton, A. Singhal, M. Mitra, and C. Buckley. 1997. Automatic text structuring and summarization. *Information Processing and Management*, 2(32).

S. Teufel and M. Moens. 1997. Sentence extraction as a classification task. In *ACL/EACL workshop on intelligent and scalable text summarization*, Madrid, Spain.

S. Wan and K. McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland.

F. Wolf and E. Gibson. 2004. Paragraph-, word-, and coherence-based approaches to sentence ranking: A comparison of algorithm and human performance. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, Barcelona, Spain, July.

L. Zhou and E. Hovy. 2003. A Web-trained extraction summarization system. In *Proceedings of Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May.

L. Zhou and E. Hovy. 2005. Digesting virtual "geek" culture: The summarization of technical internet relay chats. In *Proceedings of Association for Computational Linguistics (ACL 2005)*, Ann Arbor.

L. Zhuang, F. Jing, and X.Y. Zhu. 2006. Movie review mining and summarization. In *Proceedings of the ACM international conference on Information and knowledge management (CIKM 2006)*, Arlington, Virginia.

Part-of-speech Tagging for Middle English through Alignment and Projection of Parallel Diachronic Texts

Taesun Moon and Jason Baldridge
Department of Linguistics
University of Texas at Austin
1 University Station B5100
Austin, TX 78712-0198 USA
tsmoon, jbaldrid@mail.utexas.edu

Abstract

We demonstrate an approach for inducing a tagger for historical languages based on existing resources for their modern varieties. Tags from Present Day English source text are projected to Middle English text using alignments on parallel Biblical text. We explore the use of multiple alignment approaches and a bigram tagger to reduce the noise in the projected tags. Finally, we train a maximum entropy tagger on the output of the bigram tagger on the target Biblical text and test it on tagged Middle English text. This leads to tagging accuracy in the low 80's on Biblical test material and in the 60's on other Middle English material. Our results suggest that our bootstrapping methods have considerable potential, and could be used to semi-automate an approach based on incremental manual annotation.

1 Introduction

Annotated corpora of historical texts provide an important resource for studies of syntactic variation and change in diachronic linguistics. For example, the Penn-Helsinki Parsed Corpus of Middle English (PPCME) (Kroch and Taylor, 2000) has been used to show the existence of syntactic dialectal differences between northern and southern Middle English (Kroch et al., 2000) and to examine the syntactic evolution of the English imperative construction (Han, 2000). However, their utility rests on their having coverage of a significant amount of annotated material from which to draw patterns for such studies, and creating resources such as the PPCME require significant time and cost to produce. Corpus linguists interested in diachronic language studies thus need efficient ways to produce such resources.

One approach to get around the annotation bottleneck is to use semi-automation. For example, when producing part-of-speech tags for the Tycho Brahe corpus of Historical Portuguese (Britto et al., 2002), a set of seed sentences was manually tagged, and the Brill tagger (Brill, 1995) was then trained on those and consequently used to tag other sentences. The output was inspected for errors, the tagger was retrained and used again to tag new sentences, for several iterations.

We also seek to reduce the human effort involved in producing part-of-speech tags for historical corpora. However, our approach does so by leveraging existing resources for a language's modern varieties along with parallel diachronic texts to produce accurate taggers. This general technique has worked well for bilingual bootstrapping of language processing resources for one language based on already available resources from the other. The first to explore the idea were Yarowsky and Ngai (2001), who induced a part-of-speech tagger for French and base noun phrase detectors for French and Chinese via transfer from English resources. They built a highly accurate POS tagger by labeling English text with an existing tagger (trained on English resources), aligning that text with parallel French, projecting the automatically assigned English POS tags across these alignments, and then using the automatically labeled French text to train a new French tagger. This tech-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 390–399, Prague, June 2007. ©2007 Association for Computational Linguistics

nique has since been used for other languages and tasks, e.g. morphological analysis (Yarowsky et al., 2001), fine-grained POS tagging for Czech (Drábek and Yarowsky, 2005), and tagging and inducing syntactic dependencies for Polish (Ozdowska, 2006).

This methodology holds great promise for producing tools and annotated corpora for processing diachronically related language pairs, such as Modern English to Middle or Old English. Historical languages suffer from a paucity of machine readable text, inconsistencies in orthography, and grammatical diversity (in the broadest sense possible). This diversity is particularly acute given that diachronic texts of a given language encompass texts and genres spanning across centuries or millenia with a plethora of extra-linguistic influences to complicate the data. Furthermore, even in historically contemporaneous texts, possible dialectal variations further amplify the differences in already idiosyncratic orthographies and syntactic structure.

The present study goes further than Britto et al. (2002) by fully automating the alignment, POS tag induction, and noise elimination process. It is able to utilize the source language to a greater degree than the previously mentioned studies that attempted language neutrality; that is, it directly exploits the genetic similarity between the source and target language. Some amount of surface structural similarity between a diachronic dialect and its derivatives is to be expected, and in the case of Middle English and Modern English, such similarities are not negligible.

The automation process is further aided through the use of two versions of the Bible, which obviates the need for sentence alignment. The modern Bible is tagged using the C&C maximum entropy tagger (Curran and Clark, 2003), and these tags are transferred from source to target through high-confidence alignments aquired from two alignment approaches. A simple bigram tagger is trained from the resulting target texts and then used to relabel the same texts as Middle English training material for the C&C tagger. This tagger utilizes a rich set of features and a wider context, so it can exploit surface similarities between the source and target language. By training it with both the original (Modern English) Penn Treebank Wall Street Journal (WSJ) material and our automatically tagged Middle English Wycliffe material, we achieve an accuracy of 84.8% on predicting coarse tags, improving upon a 63.4% baseline of training C&C on the WSJ sentences alone. Furthermore, we show that the bootstrapped tagger greatly reduces the error rate on out-of-domain, non-Biblical Middle English texts.

2 Data

English provides an ideal test case for our study because of the existence of publically accessible diachronic texts of English and their translations in electronic format and because of the availability of the large, annotated Penn-Helsinki Parsed Corpus of Middle English. The former allows us to create a POS tagger via alignment and projection; the latter allows us to evaluate the tagger on large quantities of human-annotated tags.

2.1 The Bible as a parallel corpus

We take two versions of the Bible as our parallel corpus. For modern English, we utilize the NET Bible[1]. For Middle English (ME), we utilize John Wycliffe's Bible[2]. The first five lines of Genesis in both Bibles are shown in Figure 1.

The Bible offers some advantages beyond its availability. All its translations are numbered, facilitating assessment of accuracy for sentence alignment models. Also, the Bible is quite large for a single text: approximately 950,000 words for Wycliffe's version and 860,000 words for the NET bible. Finally, Wycliffe's Bible was released in the late 14th century, a period when the transition of English from a synthetic to analytical language was finalized. Hence, word order was much closer to Modern English and less flexible than Old English; also, nominal case distinctions were largely neutralized, though some verbal inflections such as distinctions for the first and second person singular in the present tense were still in place (Fennell, 2001). This places Wycliffe's Bible as far back as possible without introducing extreme nominal and verbal inflections in word alignment.

The two Bibles were cleaned and processed for the present task and then examined for levels of correspondence. The two texts were compared for

[1] The New English Translation Bible, which may be downloaded from http://www.bible.org/page.php?page_id=3086.

[2] Available for download at: http://wesley.nnu.edu/biblical_studies/wycliffe.

1 In the beginning God created the heavens and the earth.
2 Now the earth was without shape and empty, and darkness was over the surface of the watery deep, but the Spirit of God was moving over the surface of the water.
3 God said, "Let there be light." And there was light!
4 God saw that the light was good, so God separated the light from the darkness.
5 God called the light day and the darkness night. There was evening, and there was morning, marking the first day.

1 In the bigynnyng God made of nouyt heuene and erthe.
2 Forsothe the erthe was idel and voide, and derknessis weren on the face of depthe; and the Spiryt of the Lord was borun on the watris.
3 And God seide, Liyt be maad, and liyt was maad.
4 And God seiy the liyt, that it was good, and he departide the liyt fro derknessis; and he clepide the liyt,
5 dai, and the derknessis, nyyt. And the euentid and morwetid was maad, o daie.

Figure 1: The first five verses of Genesis the NET Bible (top) and Wycliffe's Bible (below).

whether there were gaps in the chapters and whether one version had more chapters over the other. If discrepancies were found, the non-corresponding chapters were removed. Next, because we assume sentences are already aligned in our approach, discrepancies in verses between the two Bibles were culled. A total of some two hundred lines were removed from both Bibles. This processing resulted in a total of 67 books[3], with 920,000 words for the Wycliffe Bible and 840,000 words for the NET Bible.

2.2 The Penn-Helsinki Parsed Corpus of Middle English

The Penn-Helsinki Parsed Corpus of Middle English is a collection of text samples derived from manuscripts dating 1150–1500 and composed during the same period or earlier. It is based on and expands upon the Diachronic Part of the Helsinki Corpus of English Texts. It contains approximately 1,150,000 words of running text from 55 sources. The texts are provided in three forms: raw, POS tagged, and parsed.

Among the texts included are portions of the Wycliffe Bible. They comprise partial sections of *Genesis* and *Numbers* from the Old Testament and *John I.1–XI.56* from the New Testament. In total, the sections of Wycliffe annotated in PPCME have some 25,000 words in 1,845 sentences. This was used as part of the test material. It is important to note that there are significant spelling differences from the full Wycliffe text that we use for alignment – this is a common issue with early writings that makes building accurate taggers for them more difficult than for the clean and consistent, edited modern texts typically used to evaluate taggers.

2.3 Tagsets

The PPCME uses a part-of-speech tag set that has some differences from that used for the Penn Treebank, on which modern English taggers are generally trained. It has a total of 84 word tags compared to the widely used Penn Treebank tag set's 36 word tags.[4] One of the main reasons for the relative diversity of the PPCME tag set is that it maintains distinctions between the *do*, *have*, and *be* verbs in addition to non-auxiliary verbs. The tag set is further complicated by the fact that composite POS tags are allowed as in *another_D+OTHER*, *midnyght_ADJ+N*, or *armholes_N+NS*.

To measure tagging accuracy, we consider two different tag sets: PTB, and COARSE. A measurement of accuracy is not possible with a direct comparison to the PPMCE tags since our approach la-

[3] 66 books shared by the churches and one book from the Apocrypha. A comparison of the two Bibles revealed that the NET Bible contained the Apocrypha, but only Baruch was shared between the two versions.

[4] In our evaluations, we collapse the many different punctuation tags down to a single tag, $PUNC$.

bels target text in Middle English with tags from the Penn Treebank. Therefore, with PTB, all non-corresponding PPCME tags were conflated if necessary and mapped to the Penn Treebank tag set. Between the two sets, only 8 tags, EX, FW, MD, TO, VB, VBD, VBN, VBP, were found to be fully identical. In cases where tags from the two sets denoted the same category/subcategory, one was simply mapped to the other. When a PPCME tag made finer distinctions than a related Penn tag and could be considered a subcategory of that tag, it was mapped accordingly. For example, the aforementioned auxiliary verb tags in the PPMCE were all mapped to corresponding subcategories of the larger VB tag group, a case in point being the mapping of the perfect participle of *have*_HVN to VBN, a plain verbal participle. For COARSE, the PTB tags were even further reduced to 15 category tags,[5] which is still six more than the core consensus tag set used in Yarkowsky and Ngai (2001). Specifically, COARSE was measured by comparing the first letter of each tag. For example, *NN* and *NNS* are conflated to *N*.

2.4 Penn Treebank Release 3

The POS tagged Wall Street Journal, sections 2 to 21, from the Penn Treebank Release 3 (Marcus et al., 1994) was used to train a Modern English tagger to automatically tag the NET Bible. It was also used to enhance the maximum likelihood estimates of a bigram tagger used to label the target text.

3 Approach

Our approach involves three components: (1) projecting tags from Modern English to Middle English through alignment; (2) training a bigram tagger; and (3) bootstrapping the C&C tagger on Middle English texts tagged by the bigram tagger. This section describes these components in detail.

3.1 Bootstrapping via alignment

Yarowsky and Ngai (2001) were the first to propose the use of parallel texts to bootstrap the creation of taggers. The approach first requires an alignment to be induced between the words of the two texts; tags are then projected from words of the source language to words of the target language. This naturally leads to the introduction of noise in the target language tags. Yarowsky and Ngai deal with this by (a) assuming that each target word can have at most two tags and interpolating the probability of tags given a word between the probabilities of the two most likely tags for that word and (b) interpolating between probabilities for tags projected from 1-to-1 alignments and those from 1-to-n alignments. Each of these interpolated probabilities is parameterized by a single variable; however, Yarowsky and Ngai do not provide details for how the two parameter values were determined/optimized.

Here, we overcome much of the noise by using two alignment approaches, one of which exploits word level similarities (present in genetically derived languages such as Middle English and Present Day English) and builds a bilingual dictionary between them. We also fill in gaps in the alignment by using a bigram tagger that is trained on the noisy tags and then used to relabel the entire target text.

The C&C tagger (Curran and Clark, 2003) was trained on the Wall Street Journal texts in the Penn Treebank and then used to tag the NET Bible (the source text). The POS tags were projected from the source to the Wycliffe Bible based on two alignment approaches, the Dice coefficient and Giza++, as described below.

3.1.1 Dice alignments

A dictionary file is built using the variation of the Dice Coefficient (Dice (1945)) used by Kay and Röscheisen (1993):

$$D(v, w) = \frac{2c}{N_A(v) + N_B(w)} \geq \theta$$

Here, c is the number of cooccurring positions and $N_T(x)$ is the number of occurrences of word x in corpus T. c is calculated only once for redundant occurrences in an aligned sentence pair. For example, it is a given that *the* will generally occur more than once in each aligned sentence. However, even if *the* occurs more than once in each of the sentences in aligned pair s_A and s_B, c is incremented only once. v and w are placed in the word alignment table if they exceed the threshold value θ, which is an empirically determined, heuristic measure.

[5] Namely, adjective, adverb, cardinal number, complementizer/preposition, conjunction, determiner, existential *there*, foreign word, interjection, infinitival *to*, modal, noun, pronoun, verb, and *wh*-words.

The dictionary was structured to establish a surjective relation from the target language to the source language. Therefore, no lexeme in the Wycliffe Bible was matched to more than one lexeme in the NET Bible. The Dice Coefficient was modified so that for a given target word v

$$D_v = \arg\max_w D(v, w)$$

would be mapped to a corresponding word from the source text, such that the Dice Coefficient would be maximized. Dictionary entries were further culled by removing (v, w) pairs whose maximum Dice Coefficient was lower than the θ threshold, for which we used the value 0.5. Finally, each word which had a mapping from the target was sequentially mapped to a majority POS tag. For example, the word *like* which had been assigned four different POS tags, IN, NN, RB, VB, by the C&C tagger in the NET Bible was only mapped to IN since the pairings of the two occurred the most frequently. The result is a mapping from one or more target lexemes to a source lexeme to a majority POS tag. In the case of *like*, two words from the target, *as* and *lijk*, were mapped thereto and to the majority tag IN.

Later, we will refer to the Wycliffe text (partially) labeled with tags projected using the Dice coefficient as DICE_1TO1.

3.1.2 GIZA++ alignments

Giza++ (Och and Ney, 2003) was also used to derive 1-to-n word alignments between the NET Bible and the Wycliffe Bible. This produces a tagged version of the Wycliffe text which we will refer to as GIZA_1TON. In our alignment experiment, we used a combination of IBM Model 1, Model 3, Model 4, and an HMM model in configuring Giza++.

GIZA_1TON was further processed to remove noise from the transferred tag set by creating a 1-to-1 word alignment: each word in the target Middle English text was given its majority tag based on the assignment of tags to GIZA_1TON as a whole. We call this version of the tagged Wycliffe text GIZA_1TO1.

3.2 Bigram tagger

Note that because the projected tags in the Wycliffe materials produced from the alignments are incomplete, there are words in the target text which have no tag. Nonetheless, a bigram tagger can be trained from maximum likelihood estimates for the words and tag sequences which were successfully projected. This serves two functions: (1) it creates a useable bigram tagger and (2) the bigram tagger can be used to fill in the gaps so that the more powerful C&C tagger can be trained on the target text.

A bigram tagger selects the most likely tag sequence T for a word sequence W by:

$$\arg\max_T P(T|W) = P(W|T)P(T)$$

Computing these terms requires knowing the transition probabilities $P(t_i|t_{i-1})$ and the emission probabilities $P(w_i|t_i)$. We use straightforward maximum likelihood estimates from data with projected tags:

$$P(t_i|t_{i-1}) = \frac{f(t_{i-1}, t_i)}{f(t_{i-1})}$$
$$P(w_i|t_i) = \frac{f(w_i, t_i)}{f(t_i)}$$

Estimates for unseen events were obtained through add-one smoothing.

In order to diversify the maximum likelihood estimates and provide robustness against the errors of any one alignment method, we concatenate several tagged versions of the Wycliffe Bible with tags projected from each of our methods (DICE_1TO1, GIZA_1TON, and GIZA_1TO1) and the NET Bible (and its tags from the C&C tagger).

3.3 Training C&C on projected tags

The bigram tagger learned from the aligned text has very limited context and cannot use rich features such as prefixes and suffixes of words in making its predictions. In contrast, the C&C tagger, which is based on that of Ratnaparkhi (1996), utilizes a wide range of features and a larger contextual window including the previous two tags and the two previous and two following words. However, the C&C tagger cannot train on texts which are not fully tagged for POS, so we use the bigram tagger to produce a completely labeled version of the Wycliffe text and train the C&C tagger on this material. The idea is that even though it is training on imperfect material, it will actually be able to correct many errors by virtue of its greater discriminitive power.

	Evaluate on PPCME Wycliffe		Evaluate on PPCME Test	
Model	PTB	COARSE	PTB	COARSE
(a) Baseline, tag NN	9.0	17.7	12.6	20.1
(b) C&C, trained on gold WSJ	56.2	63.4	56.2	62.3
(c) Bigram, trained on DICE_1TO1 and GIZA_1TON	68.0	73.1	43.9	49.8
(d) Bigram, trained on DICE_1TO1 and GIZA_1TO1	74.8	80.5	58.0	63.9
(e) C&C, trained on BOOTSTRAP (920k words)	78.8	84.1	61.3	67.8
(f) C&C, trained on BOOTSTRAP and WSJ and NET	79.5	84.8	61.9	68.5
(g) C&C, trained on (gold) PPCME Wycliffe (25k words)	n/a	n/a	71.0	76.0
(h) C&C, trained on (gold) PPCME training set (327k words)	95.9	96.9	93.7	95.1

Figure 2: Tagging results. See section 4 for discussion.

We will refer to the version of the Wycliffe text (fully) tagged in this way as BOOTSTRAP.

4 Experiments

The M3 and M34 subsections[6] of the Penn Helsinki corpus were chosen for testing since it is not only from the same period as the Wycliffe Bible but since it also includes portions of the Wycliffe Bible. A training set of 14 texts comprising 330,000 words was selected to train the C&C tagger and test the cost necessary to equal or exceed the automatic implementation. The test set consists of 4 texts with 110,000 words. The sample Wycliffe Bible with the gold standard tags has some 25,000 words.

The results of the various configurations are given in Figure 2, and are discussed in detail below.

4.1 Baselines

We provide two baselines. The first is the result of giving every word the common tag NN. The second baseline was established by directly applying the C&C tagger, trained on the Penn Treebank, to the PPCME data. The results are given in lines (a) and (b) of Figure 2 for the first and second baselines, respectively. As can be seen, the use of the Modern English tagger already provides a strong starting point for both evaluation sets.

4.2 Bigram taggers

In section 3.1, we discuss three versions of the Wycliffe target text labeled with tags projected across alignments from the NET Bible. The most straightforward of these were DICE_1TO1 and GIZA_1TON which directly use the alignments from the methods. Training a bigram tagger on these two sources leads to a large improvement over the C&C baseline on the PPCME Wycliffe sentences, as can be seen by comparing line (c) to line (b) in Figure 2. However, performance drops on the PPCME Test sentences, which come from different domains than the bigram tagger's automatically produced Wycliffe training material. This difference is likely to do good estimates of $P(w_i|t_i)$, but poor estimates of $P(t_i|t_{i-1})$ due to the noise introduced in GIZA_1TON.

More conservative tags projection is thus likely to have a large effect on the out-of-domain performance of the learned taggers. To test this, we trained a bigram tagger on DICE_1TO1 and the more conservative GIZA_1TO1 projection. This produces further gains for the PPCME Wycliffe, and enormous improvements on the PPCME Test data (see line (d) of Figure 2). This result confirms that conservativity beats wild guessing (at the risk of reduced coverage) for bootstrapping taggers in this way. This is very much in line with the methodology of Yarowksy and Ngai (2001), who project a small number of tags out of all those predicted by alignment. They achieve this restriction by directly adjusting the probabality mass assigned to projected tags; we do it by using two versions of the target text with tags projected in

[6] Composition dates and manuscript dates for M3 are 1350-1420. The composition dates for M34 are the same but the manuscripts date 1420-1500

two different 1-to-1 ways.

4.3 Bootstrapping the C&C tagger

As described in section 3.3, a bigram tagger trained on DICE_1TO1 and GIZA_1TO1 (i.e., the tagger of line (d)), was used to relabel the entire Wycliffe target text to produce training material for C&C, which we call BOOTSTRAP. The intention is to see whether the more powerful tagger can bootstrap off imperfect tags and take advantage of its richer features to produce a more accurate tagger. As can be seen in row (e) of Figure 2, it provides a 3-4% gain across the board over the bigram tagger which produced its training material (row (d)).

We also considered whether using all available (non-PPCME) training material would improve tagging accuracy by training C&C on BOOTSTRAP, the Modern English Wall Street Journal (from the Penn Treebank), and the automatically tagged NET text[7] It did produce slight gains on both test sets over C&C trained on BOOTSTRAP alone. This is likely due to picking up some words that survived unchanged to the Modern English. Of course, the utility of modern material used directly in this manner will likely vary a great deal depending on the distance between the two language variants. What is perhaps most interesting is that adding the modern material did not *hurt* performance.

4.4 Upperbounds

It is apparent from the results that there is a strong domain effect on the performance of both the bigram and C&C taggers which have been trained on automatically projected tags. There is thus a question of how well we could ever hope to perform on PPCME Test given perfect tags from the Wycliffe texts. To test this, C&C was trained on the *PPCME* version of Wycliffe, which has human annotated standard tags, and then applied on the PPCME test set. We also compare this to training on PPCME texts which are similar to those in PPCME Test.

The results, given in lines (g) and (h) of Figure 2, indicate that there is a likely performance cap on non-Biblical texts when bootstrapping from parallel Biblical texts. The results in line (h) also show that the non-Biblical texts are more difficult, even with gold training material. This is likely due to the wide variety of authors and genres contained in these texts – in a sense, everything is slightly out-of-domain.

4.5 Learning curves with manual annotation

The upperbounds raise two questions. One is whether the performance gap between (g) and (h) in Figure 2 on PPCME Test is influenced by the significant difference in the size of their training sets. The other is how much gold-standard PPCME training material would be needed to match the performance of our best bootstrapped tagger (line (f)). This is a natural question to ask, as it hits at the heart of the utility of our essentially unsupervised approach versus annotating target texts manually.

To examine the cost of manually annotating the target language as compared to our unsupervised method, the C&C tagger was also trained on randomly selected sets of sentences from PPCME (disjoint from PPCME Test). Accuracy was measured on PPCME Wycliffe and Test for a range of training set sizes, sampled at exponentially increasing values (25, 50, 100, . . . , 12800). Though we trained on and predicted the full tagset used by the PPCME, it was evaluated on PTB to give an accurate comparison.[8]

The learning curves on both test sets are shown in Figure 3. The accuracy of the C&C tagger increases rapidly, and the accuracy exceeds our automated method on PPCME Test with just 50 labeled sentences and on the PPCME Wycliffe with 400 examples. This shows the domain of the target text is served much better with the projection approach.

To see how much gold-standard PPCME Wycliffe material is necessary to beat our best bootstrapped tagger, we trained the tagger as in (g) of Figure 2 with varying amounts of material. Roughly 600 labeled sentences were required to beat the performance of 61.9%/68.5% (line (f), on both metrics).

These learning curves suggest that when the domain for which one wishes to produce a tagger is significantly different from the aligned text one has available (in this and in many cases, the Bible), then labeling a small number of examples by hand is a quite reasonable approach (provided random sampling is used). However, if one is not careful, considerable effort could be put into labeling sentences

[7] This essentially is partial self-training since C&C trained on WSJ was used to produce the NET tags.

[8] Evaluation with the full PPCME set produces accuracy figures about 1% lower.

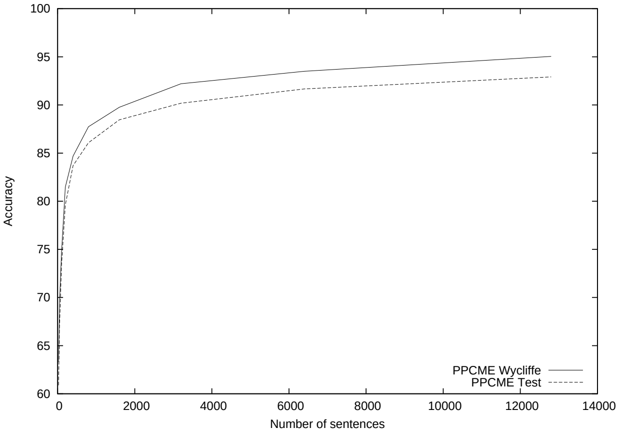


Figure 3: Learning curve showing the accuracy for PTB tags of the C&C tagger on both Bible and Test as it is given more gold-standard PPCME training sentences.

that are not optimal overall (imagine getting unlucky and starting out by manually annotating primarily Wycliffe sentences). The automated methods we present here start producing good taggers immediately, and there is much room for improving them further. Additionally, they could be used to aid manual annotation by proposing high-confidence labels even before any annotation has begun.

5 Related work

Despite the fact that the Bible has been translated into many languages and that it constitutes a solid source for studies in NLP with a concentration on machine translation or parallel text processing, the number of studies involving the Bible is fairly limited. A near exhaustive list is Chew et al.(2006), Melamed(1998), Resnik et al.(1999), and Yarowsky et al.(2001).

Yarowsky and Ngai (2001) is of central relevance to this study. The study describes an unsupervised method for inducing a monolingual POS tagger, base noun-phrase bracketer, named-entity tagger and morphological analyzers from training based on parallel texts, among many of which the Bible was included. This is particularly useful given that no manually annotated data is necessary in the target language and that it works for two languages from different families such as French and Chinese. In the case of POS tagging, only the results for English-French are given and an accuracy of 96% is achieved. Even though this accuracy figure is based on a reduced tag set smaller than the COARSE used in this study, it is still a significant increase over that achieved here. However, their method had the advantage of working in a domain that overlaps with the training data for their POS tagger. Second, the the French tag set utilized in that study is considerably smaller than the Penn Helsinki tag set, a possible source of greater noise due to its size.

Drábek and Yarowsky (2005) create a fine-grained tagger for Czech and French by enriching the tagset for parallel English text with additional morphological information, which, though not directly attested by the impoverished English morphological system (e.g. number on adjectives), typically does appear in other languages.

6 Conclusion

The purpose of the study was to implement a POS tagger for diachronic texts of maximal accuracy with minimal cost in terms of labor, regardless of the shortcuts taken. Such taggers are the building blocks in the design of higher level tools which depend on POS data such as morphological analyzers and parsers, all of which are certain to contribute to diachronic language studies and genetic studies of language change.

We showed that using two conservative methods for projecting tags through alignment significantly improves bigram POS tagging accuracies over a baseline of applying a Modern English tagger to Middle English text. Results were improved further by training a more powerful maximum entropy tagger on the predictions of the bootstrapped bigram tagger, and we observed a further, small boost by using Modern English tagged material in addition to the projected tags when training the maximum entropy tagger.

Nonetheless, our results show that there is still much room for improvement. A manually annotated training set of 400–800 sentences surpassed our best bootstrapped tagger. However, it should be noted that the learning curve approach was based on domain neutral, fully randomized, incremental texts, which are not easily replicated in real world applications. The domain effect is particularly evident in training on the sample Wycliffe and tagging on the test PPCME set. Of course, our approach can be integrated with one based on annotation by using our bootstrapped taggers to perform semi-automated annotation, even *before* the first human-annotated tag has been labeled.

It is not certain how our method would fare on the far more numerous parallel diachronic texts which do not come prealigned. It is also questionable whether it would still be robust on texts predating Middle English, which might as well be written in a foreign language when compared to Modern English. These are all limitations that need to be explored in the future.

Immediate improvements can be sought for the algorithms themselves. By restricting the mapping of words to only one POS tag in the Wycliffe Bible, this seriously handicapped the utility of a bigram tagger. It should be relatively straightforward to transfer the probability mass of multiple POS tags in a modern text to corresponding words in a diachronic text and include this modified probability in the bigram tagger. When further augmented for automatic parameter adjustment with the forward-backward algorithm, accuracy rates might increase further. Furthermore, different algorithms might be better able to take advantage of similarities in orthography and syntactic structure when constructing word alignment tables. Minimum Edit Distance algorithms seem particularly promising in this regard.

Finally, it is evident that the utility of the Bible as a potential resource of parallel texts has largely gone untapped in NLP research. Considering that it has probably been translated into more languages than any other single text, and that this richness of parallelism holds not only for synchrony but diachrony, its usefulness would apply not only to the most immediate concern of building language tools for many of the the world's underdocumented languages, but also to cross-linguistic studies of unprecedented scope at the level of language genera. This study shows that despite the fact that any two Bibles are rarely in a direct parallel relation, standard NLP methods can be applied with success.

References

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Helena Britto, Marcelo Finger, and Charlotte Galves, 2002. *Computational and linguistic aspects of the construction of The Tycho Brahe Parsed Corpus of Historical Portuguese*. Tübingen: Narr.

Peter A. Chew, Steve J. Verzi, Travis L. Bauer, and Jonathan T. McClain. 2006. Evaluation of the bible as a resource for cross-language information retrieval. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability, Sydney, July 2006*, pages 68–74.

James R Curran and Stephen Clark. 2003. Investigating gis and smoothing for maximum entropy taggers. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-03)*.

Lee R. Dice. 1945. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302.

Elliott Franco Drábek and David Yarowsky. 2005. Induction of fine-grained part-of-speech taggers via classifier combination and crosslingual projection. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 49–56, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Barbara A. Fennell. 2001. *A History of English: A Sociolinguistic Approach*. Blackwell, Oxford.

Chung-Hye Han, 2000. *The Evolution of Do-Support In English Imperatives*, pages 275–295. Oxford University Press.

Martin Kay and Martin Röscheisen. 1993. Text-translation alignment. *Computational Linguistics*, 19(1):121–142.

Anthony Kroch and Ann Taylor. 2000. Penn-helsinki parsed corpus of middle english, second edition.

Anthony Kroch, Ann Taylor, and Donald Ringe. 2000. The middle english verb-second constraint: A case study in language contact and language change. *Amsterdam Studies in the Theory and History of Linguistic Science Series*, 4:353–392.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Dan I. Melamed. 1998. Manual annotation of translation equivalence: The blinker project. In *Technical Report 98-07, Institute for Research in Cognitive Science, Philadelphia*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Sylwia Ozdowska. 2006. Projecting pos tags and syntactic dependencies from english and french to polish in aligned corpora. In *EACL 2006 Workshop on Cross-Language Knowledge Induction*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey.

Philip Resnik, Mari Broman Olsen, and Mona Diab. 1999. The bible as a parallel corpus: Annotating the ”book of 2000 tongues”. *Computers and the Humanities*, 33(1–2):129–153.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Appendix

Figure 4 provides the full mapping from PPCME tags to the Penn Treebank Tags used in our evaluation.

PPCME→PTB	PPCME→PTB
ADJR→JJR	N→NN
ADJS→JJS	N$→NN
ADV→RB	NEG→RB
ADVR→RBR	NPR→NNP
ADVS→RBS	NPR$→NNP
ALSO→RB	NPRS→NNPS
BAG→VBG	NPRS$→NNPS
BE→VB	NS→NNS
BED→VBD	NS$→NNS
BEI→VB	NUM→CD
BEN→VBN	NUM$→CD
BEP→VBZ	ONE→PRP
C→IN	ONE$→PRP$
CODE→CODE	OTHER→PRP
CONJ→CC	OTHER$→PRP
D→DT	OTHERS→PRP
DAG→VBG	OTHERS$→PRP
DAN→VBN	P→IN
DO→VB	PRO→PRP
DOD→VBD	PRO$→PRP$
DOI→VB	Q→JJ
DON→VBN	Q$→JJ
DOP→VBP	QR→RBR
E_S→E_S	QS→RBS
ELSE→RB	RP→RB
EX→EX	SUCH→RB
FOR→IN	TO→TO
FOR+TO→IN	VAG→VBG
FP→CC	VAN→VBN
FW→FW	VB→VB
HAG→VBG	VBD→VBD
HAN→VBN	VBI→VB
HV→VB	VBN→VBN
HVD→VBD	VBP→VBP
HVI→VB	WADV→WRB
HVN→VBN	WARD→WARD
HVP→VBP	WD→WDT
ID→ID	WPRO→WP
INTJ→UH	WPRO$→WP$
MAN→PRP	WQ→IN
MD→MD	X→X
MD0→MD	

Figure 4: Table of mappings from PPCME tags to Penn Treebank Tags.

Flexible, Corpus-Based Modelling of Human Plausibility Judgements

Sebastian Padó and **Ulrike Padó**
Computational Linguistics
Saarland University
Saarbrücken, Germany
{pado,ulrike}@coli.uni-sb.de

Katrin Erk
Dept. of Linguistics
University of Texas at Austin
Austin, Texas
katrin.erk@mail.utexas.edu

Abstract

In this paper, we consider the computational modelling of human plausibility judgements for verb-relation-argument triples, a task equivalent to the computation of selectional preferences. Such models have applications both in psycholinguistics and in computational linguistics.

By extending a recent model, we obtain a completely corpus-driven model for this task which achieves significant correlations with human judgements. It rivals or exceeds deeper, resource-driven models while exhibiting higher coverage. Moreover, we show that our model can be combined with deeper models to obtain better predictions than from either model alone.

1 Introduction

One fundamental and intuitive finding in experimental psycholinguistics is that humans judge the plausibility of a verb-argument pair vastly differently depending on the semantic relation in the pair. Table 1 lists example human judgements which McRae et al. (1998) elicited by asking about the plausibility of, e.g., a hunter shooting (relation *agent*) or being shot (relation *patient*). McRae et al. found that "hunter" is judged to be a very plausible *agent* of "shoot" and an implausible *patient*, while the reverse is true for "deer". In linguistics, this phenomenon is explained by *selectional preferences* on verbs' argument positions; we use *plausibility* and *fit with selectional preferences* interchangeably.

Verb	Relation	Noun	Plausibility
shoot	agent	hunter	6.9
shoot	patient	hunter	2.8
shoot	agent	deer	1.0
shoot	patient	deer	6.4

Table 1: Verb-relation-noun triples with plausibility judgements on a 7-point scale (McRae et al., 1998)

In this paper, we consider computational models that predict human plausibility ratings, or the fit of selectional preferences and argument, for such $(verb, relation, argument)$, in short, (v, r, a), triples. Being able to model this type of data is relevant in a number of ways. From the point of view of psycholinguistics, selectional preferences have an important effect in human sentence processing (e.g., McRae et al. (1998), Trueswell et al. (1994)), and models of selectional preferences are therefore necessary to inform models of this process (Padó et al., 2006). In computational linguistics, a multitude of tasks is sensitive to selectional preferences, such as the resolution of ambiguous attachments (Hindle and Rooth, 1993), word sense disambiguation (McCarthy and Carroll, 2003), semantic role labelling (Gildea and Jurafsky, 2002), or testing the applicability of inference rules (Pantel et al., 2007).

A number of approaches has been proposed to model selectional preference data (Padó et al., 2006; Resnik, 1996; Clark and Weir, 2002; Abe and Li, 1996). These models generally operate by generalising from seen (v, r, a) triples to unseen ones. By relying on resources like corpora with semantic role annotation or the WordNet ontology, these models

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 400–409, Prague, June 2007. ©2007 Association for Computational Linguistics

generally share two problems: (a), limited coverage; and (b), the resource (at least partially) predetermines the generalisations that they can make.

In this paper, we investigate whether it is possible to predict the plausibility of (v, r, a) triples in a completely corpus-driven way. We build on a recent selectional preference model (Erk, 2007) that bases its generalisations on word similarity in a vector space. While that model relies on corpora with semantic role annotation, we show that it is possible to predict plausibility ratings solely on the basis of a parsed corpus, by using shallow cues and a suitable vector space specification.

For evaluation, we use two balanced data sets of human plausibility judgements, i.e., datasets where each verb is paired both with a good agent and a good patient, and where both nouns are presented in either semantic relation (as in Table 1). Using balanced test data is a particularly difficult task, since it forces the models to account reliably both for the influence of the semantic relation (*agent/patient*) and of the argument head ("hunter"/"deer").

We obtain three main results: (a), our model is able to match the superior performance of the model proposed by Padó et al. (2006), while retaining the high coverage of the model proposed by Resnik (1996); (b), using parsing as a preprocessing step improves the model's performance significantly; and (c), a combination of our model with the Padó model exceeds both individual models in accuracy.

Plan of the paper. In Section 2, we give an overview of existing selectional preferences and vector space models. Section 3 introduces our model and discusses its parameters. Sections 4 and 5 present our experimental setup and results. Section 6 concludes.

2 Related Work

Modelling Selectional Preferences with Grammatical Functions. The idea of inducing selectional preferences from corpora was introduced by Resnik (1996). He approximated the semantic verb-argument relations in (v, r, a) triples by grammatical functions, which are readily available for large training corpora. His basic two-step procedure was followed by all later approaches: (1), extract argument headwords for a given predicate and relation from a corpus; (2), generalise to other, similar words using the WordNet noun hierarchy. Other models also relying on the WordNet resource include Abe and Li (1996) and Clark and Weir (2002).

We present Resnik's model in some detail, since we will use it for comparison below. Resnik first computes the overall selectional preference strength for each verb-relation pair, i.e. the degree of "constrainedness" of each relation. This quantity is estimated as the difference (in terms of the Kullback-Leibler divergence D) between the distribution over WordNet argument classes given the relation, $p(c|r)$, and the distribution of argument classes given the current verb-relation combination, $p(c|v, r)$. The intuition is that a verb-relation pair that only allows for a limited range of argument heads will have a probability distribution over argument classes that strongly diverges from the prior distribution.

Next, the selectional association of the triple, $A(v, r, c)$, is computed as the ratio of the selectional preference strength for this particular class, divided by the overall selectional preference strength of the verb-relation pair. This is shown in Equation 1.

$$A(v, r, c) = \frac{p(c|v, r) log \frac{p(c|v,r)}{p(c|r)}}{D(p(c|r)||p(c|v, r))} \qquad (1)$$

Finally, the selectional preference between a verb, a relation, and an argument head is taken to be the selectional association of the verb and relation with the most strongly associated WordNet ancestor class of the argument.

WordNet-based approaches however face two problems. One is a coverage problem due to the limited size of the resource (see the task-based evaluation in Gildea and Jurafsky (2002)). The other is that the shape of the WordNet hierarchy determines the generalisations that the models make. These are not always intuitive. For example, Resnik (1996) observes that $(answer, obj, tragedy)$ receives a high preference because "tragedy" in WordNet is a type of written communication, which is a preferred argument class of "answer".

Rooth et al. (1999) present a fundamentally different approach to selectional preference induction which uses soft clustering to form classes for generalisation and does not take recourse to any hand-crafted resource. We will argue in Section 6 that our model allows more control over the generalisations made.

Modelling Selectional Preferences with Thematic Roles. Padó et al. (2006) present a deeper model for the plausibility of (v, r, a) triples that approximates the relations with thematic roles. It estimates the selectional preferences of a verb-role pair with a generative probability model that equates the plausibility of a (v, r, a) triple with the joint probability of seeing the thematic role with the verb-argument pair. In addition, the model also considers the verb's sense s and the grammatical function gf of the argument; however, since the model is generative, it can make predictions even when not all variables are instantiated. The final model is shown in Equation 2.

$$Plausibility_{v,r,a} = P(v, s, r, a, gf) \tag{2}$$

The induction of this model from the FrameNet corpus of semantically annotated training data (Fillmore et al., 2003) encounters a serious sparse data problem, which is approached by the application of word-class-based and Good-Turing re-estimation smoothing. The resulting model's plausibility predictions are significantly correlated to human judgements, but because of the use of verb-specific thematic roles, the model's coverage is still restricted by the verb coverage of the training corpus.

Vector Space Models. Another class of models that has found wide application in lexical semantics is the family of vector space models. In a vector space model, each *target word* is represented as a vector, typically constructed from co-occurrence counts with context words in a large corpus (the so-called *basis elements*). The underlying assumption is that words with similar meanings occur in similar contexts, and will be assigned similar vectors. Thus, the distance between the vectors of two target words, as given by some distance measure (e.g., Cosine or Jaccard), is a measure of their *semantic similarity*.

Vector space models are simple to construct, and the semantic similarity they provide has found a wide range of applications. Examples in NLP include information retrieval (Salton et al., 1975), automatic thesaurus extraction (Grefenstette, 1994), and predominant sense identification (McCarthy et al., 2004). In cognitive science, they have been used to account for the influence of context on human lexical processing (McDonald and Brew, 2004), and to model lexical priming (Lowe and McDonald, 2000).

A drawback of vector space models is the difficulty of interpreting what some degree of "generic semantic similarity" between two target words means in linguistic terms. In particular, this similarity is not sensitive to selectional preferences over *specific* semantic relations, and thus cannot model the plausibility data we are interested in. The next section demonstrates how the integration of ideas from selectional preference induction makes this distinction possible.

3 The Vector Similarity Model: Corpus-Based Modelling of Plausibility

3.1 Model Architecture

Our model builds on the architecture of Erk (2007). It combines the idea underlying the selectional preference models from Section 2, namely to predict plausibility by generalising over head words, with vector space similarity. The fundamental idea of our model is to model the plausibility of the triple (v, r, a) by comparing the argument head a to other headwords a' which we have already seen in a corpus for the same verb-relation pair (v, r), and which we therefore assume to be plausible. We write $\text{Seen}_r(v)$ for the *set of seen headwords*. Our intuition is that if a is similar to the words in $\text{Seen}_r(v)$, then the triple (v, r, a) is plausible; conversely, if it is very dissimilar, then the triple is implausible.

Concretely, we judge the plausibility of the triple by averaging over the similarity of the vector for a to all vectors for the seen headwords in $\text{Seen}_r(v)$:

$$Pl(v, r, a) = \sum_{a' \in \text{Seen}_r(v)} \frac{w(a') \cdot sim(a, a')}{|\text{Seen}_r(v)|} \tag{3}$$

where w is a weight factor specific to each a'. w can be used to implement different weighting schemes that encode prior knowledge, e.g., about the reliability of different words in $\text{Seen}_r(v)$. In this paper, we only consider a very simple weighting factor, namely the frequency of the seen headwords. This encodes the assumption that similarity to frequent head words is more important than similarity to infrequent ones:

$$Pl(v, r, a) = \sum_{a' \in \text{Seen}_r(v)} \frac{f(a') \cdot sim(a, a')}{|\text{Seen}_r(v)|} \tag{4}$$

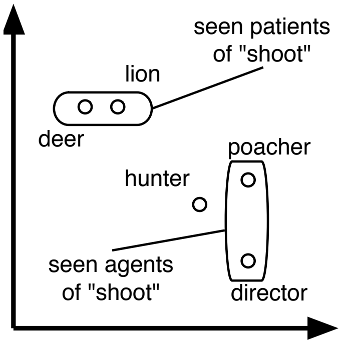


Figure 1: A vector space for estimating the plausibilities of $(shoot, agent, hunter)$ and $(shoot, patient, hunter)$.

This model can be seen as a straightforward implementation of the selectional preference induction process of generalising from seen headwords to other, similar words. By using vector space representations to judge the similarity of words, we obtain a completely corpus-driven model that does not require any additional resources and is very flexible. A complementary view on this model is as a generalisation of traditional vector space models that computes similarity not between two vectors, but between a vector and a set of other vectors. By using the vectors for seen headwords of a given relation as this set, the similarity we compute is specific to this relation.

Example. Figure 1 shows an example vector space. Consider v = “shoot”, r = *agent*, and a = “hunter”. In order to judge whether a hunter is a plausible agent of “shoot”, the vector space representation of “hunter” is compared to all representations of known agents of "shoot”, namely “poacher” and “director”. Due to the nearness of the vector for “hunter” to these two vectors, “hunter” will be judged a fairly good agent of “shoot”. Compare this with the result for the role *patient*: “hunter” is further away from “lion” and “deer”, and will therefore be found to be a rather bad patient of “shoot”. However, “hunter” is still more plausible as a patient of “shoot” than e.g., “director”.

3.2 Instantiating the Model: Unparsed vs. Parsed Corpora

The two major tasks which need to be addressed to obtain an instance of this model are (a), determining the sets of seen head words $\text{Seen}_r(v)$, and (b), the construction of a vector space. Erk (2007) extracted the set of seen head words from corpora with semantic role annotation, and used only a single vector space representation. In this paper, we eliminate the reliance on special annotation by considering shallow approximations of the semantic relations in question. In addition, we discuss in detail which properties of the vector space are crucial for the prediction of plausibility ratings, a much more fine-grained task than the pseudo-word disambiguation task presented in Erk (2007) that is more closely related to semantic role labelling. The goal of our exposition is thus to develop a model that can use more training data, and represent the corpus information optimally in order to obtain superior coverage.

In fact, tasks (a) and (b) can be solved on the basis of unparsed corpora, but we would expect the results to be rather noisy. Fortunately, the state of the art in broad-coverage (Lin, 1993) and unsupervised (Klein and Manning, 2004) dependency parsing allows us to treat dependency parsing merely as a preprocessing step. We therefore describe two instantiations of our model: one based on an unprocessed corpus, and one based on a dependency-based parsed corpus. By comparing the models, we can gauge whether syntactic preprocessing improves model performance. In the following, we describe the strategies the two models adopt for (a) and (b).

Identifying seen head words for relations. Recall that the set $\text{Seen}_r(v)$ is supposed to contain known head words a that are observed in the corpus as triples (v, r, a). In a parsed corpus, we can approximate the relation *agent* by the dependency relation of `subject` provided by the parser, and the relation *patient* by the dependency relation of `object`. In an unparsed corpus, these grammatical relations are unavailable, and the only straightforward evidence we can use is word order. In this case, we assume that words directly adjacent to the left of a predicate are subjects, and therefore *agent*s, whereas words directly to its right are objects, and thus *patient*s.

Vector space topology. The success of our method depends directly on the topology of the vector space. More specifically, two words should only be assigned similar vectors if they are in fact of similar plausibility. If this is not the case, there is no guarantee that a word a that is similar to the words in $\text{Seen}_r(v)$ forms

Target Basis elements	deer	hunter
shoot	10	10
escape	12	12

Target Basis elements	deer	hunter
shoot-SUBJ	0	8
shoot-OBJ	10	2
escape-SUBJ	10	5
escape-OBJ	2	7

Figure 2: Two vector spaces, using as basis elements either context words (above) or words paired with grammatical functions (below)

a plausible triple (v, r, a) itself (cf. Figure 1).

The topology, in turn, is related to the choice of basis elements. Traditional vector space models use context words as basis elements of the space. The top table in Figure 2 illustrates our intuition that such spaces are problematic: “deer” and “hunter” receive identical vectors, even though they show complementary plausibility ratings (cf. Table 1). The reason is that “deer” and “hunter” often co-occur quite closely to one another (e.g., in the vicinity of “shoot”), and thus show a very similar profile in terms of context words. In preliminary experiments, we found that vector spaces with context words as basis elements are in fact unable to distinguish such word pairs reliably.

In contrast, the bottom table in Figure 2 indicates that this problem can be alleviated by using context words *combined with the grammatical relation to the target word* as basis elements. Target words now receive different representations, depending on the grammatical function in which they occur with context words. In consequence, resulting spaces can distinguish, for example, between “hunter” and “deer”.

We adopt word-function pairs as basis elements for the vector spaces in all our models. In a dependency-parsed corpus, the basis elements can be directly read off the syntactic structure. In an unparsed corpus, we again fall back on word order, appending to each context word its relative position to the target word.

4 Experimental Setup

Experimental Materials. In order to make our evaluation comparable to the earlier modelling study by Padó et al. (2006), we present evaluations on the two plausibility judgement datasets used there.[1]

The first dataset consists of 100 data points from McRae et al. (1998). Our example in Table 1, which is taken from this dataset, demonstrates its *balanced* structure: 25 verbs are paired with two arguments and two relations each, such that each argument is highly plausible in one relation, but implausible in the other. The resulting distribution of ratings is thus highly bimodal. Models can only reliably predict the human ratings in this data set if they can capture the difference between verb argument slots as well as as between individual fillers.

The second, larger dataset is less strictly balanced, since its triples are constructed on the basis of corpus co-occurrences (Padó et al., 2006). 18 verbs are combined with the three most frequent subjects and objects from both the Penn Treebank and the FrameNet corpus. Each verb-argument pair was rated both as an agent and as a patient, which leads to a total of 24 rated triples per verb. The dataset contains ratings for a total of 414 triples, due to overlap between corpora. The resulting judgements show a more even distribution of ratings than the McRae data.

Vector Similarity Models. Following our exposition in the last section, we construct two instantiations of our vector similarity model, one using unparsed and one parsed data. Both are trained on the complete British National Corpus (Burnard, 1995, BNC) with more than six million sentences.

The unparsed model (Unparsed) uses the BNC without any pre-processing. We first construct the set of known headwords, $\text{Seen}_r(v)$, as follows: All words up to 2 words to the left of instances of v are assumed to be `subjects`, and thus agents; vice versa for patients to the right. Then, we construct semantic space representations for the experimental arguments and known headwords, adopting optimal parameter settings from the literature (Padó and Lapata, 2007). This means a context window of 5 words to either side and 2,000 basis elements (dimensions), which are formed by the most frequent 1,000 words

[1] We are grateful to Ken McRae for his dataset.

in the BNC, combined with each of the relations agent and patient. All counts are log-likelihood transformed (Lowe, 2001).

To construct the parsed model (Parsed), we dependency-parsed the BNC with Minipar (Lin, 1993). We first obtain the seen headwords $\text{Seen}_r(v)$ by using all subjects and objects of v as agents and patients, respectively. We then construct a vector space for the experimental arguments and known headwords.[2] We use 2,000 dimensions again, but adopt the most frequent $(\mathit{head}, \mathit{grammatical\ function})$ pairs in the BNC as basis elements. The context window is formed by `subject` and `object` dependencies. All counts are log-likelihood transformed.

We experiment with two distance measures to compute vector similarity, namely the Jaccard Coefficient and Cosine Distance, both of which have been shown to yield good performance in NLP tasks (Lee, 1999; McDonald and Lowe, 1998).

Evaluation Procedure. We evaluate our models by correlating the predicted plausibility values with the human judgements, which range between 1 and 7. Since the human judgement data is not normally distributed, we use Spearman's ρ, a non-parametric rank-order test. We determine the statistical significance of differences in correlation strength using the method described in Raghunathan (2003). This method can deal with missing values and thus allows us to compare models with different coverage.

It is difficult to specify a straightforward baseline for our correlation-based evaluation. In contrast to classification tasks, where models choose one out of a fixed number of classes, our model predicts continuous data. This task is more difficult to approximate, e.g., using frequency information.

With respect to upper bounds, we hold that automatic models of plausibility cannot be expected to surpass the typical agreement on the plausibility judgement task between human participants. Thus, we assume an upper bound of $\rho \approx 0.7$.

Comparison against Other Models. We compare our performance to two models from the literature discussed in Section 2. The first model (Pado) is the thematic role-based model by Padó et al. (2006) trained on the FrameNet (Fillmore et al., 2003) release 1.2 example sentences, a subset of the BNC annotated with semantic roles. This corpus contains about 57,000 sentences, which corresponds to roughly 1% of the BNC data.

The second model (Resnik) is the WordNet-based selectional preference model by Resnik (1996), trained on the dependency-parsed BNC (see above).

Model	Coverage	Spearman's ρ
Unparsed Cosine	90%	0.023, ns
Unparsed Jaccard	90%	0.044, ns
Parsed Cosine	91%	0.218, *
Parsed Jaccard	91%	0.129, ns
Resnik	94%	0.028, ns
Pado	56%	0.415, **

Table 2: Model performance on McRae data. *: $p < 0.05$, **: $p < 0.01$

5 Experimental Evaluation

The McRae Dataset. Table 2 summarises our results on the McRae dataset. The upper part shows the results for our two vector similarity models (Parsed/Unparsed), combined with the two distance measures (Cosine/Jaccard). The lower part shows the two resource-based models we use for comparison.

We find that all vector similarity models exhibit high coverage (above 90%), and one model (Parsed Cosine) can predict human judgements with a significant correlation. The instantiation of the model has a significant impact on the performance: The Parsed models clearly outperform the Unparsed models. The effect of the distance measure is less clear-cut, since the Unparsed models perform better with Jaccard, while the Parsed models prefer Cosine.

The deep semantic plausibility model (Pado) makes predictions only for slightly more than half of the data. This low coverage is a direct result of the small overlap in verbs between the McRae dataset and the FrameNet corpus. However, on the data points it covers, it achieves a significant correlation to human judgements. The correlation coefficient is numerically much higher than that of the Parsed Cosine model, but due to the large coverage difference, the two models are not statistically distinguishable.

[2] This space was computed using the `DependencyVectors` software described in Padó and Lapata (2007). This software can be downloaded from `http://www.coli.uni-saarland.de/~pado/dv.html`.

Model	Coverage	Spearman's ρ
Unparsed Cosine	98%	0.117, *
Unparsed Jaccard	98%	0.149, **
Parsed Cosine	98%	0.479, ***
Parsed Jaccard	98%	0.120, *
Resnik	98%	0.237, ***
Pado	97%	0.515, ***

Table 3: Model performance on Pado data. *: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$

Resnik's WordNet-based model shows a coverage that is comparable to the vector similarity models, but does not achieve a significant correlation to the human judgements.

The Pado Dataset. Table 3 summarises the results for the Pado dataset. Since all verbs in this dataset are covered in FrameNet, the deep Pado model shows a coverage comparable to all other models, at >95%.

The main difference to the McRae dataset lies in the models' performance. We find that all models, including the Unparsed vector models and Resnik, manage to achieve significant correlations with the human judgements. Within the vector similarity models, the same trends hold as for the McRae dataset: Parsed outperforms Unparsed, and the best combination is Parsed Cosine. The models fall into two clearly separated groups: The Pado and Parsed Cosine models achieve a highly significant correlation, and are statistically indistinguishable. They significantly outperform the second group ($p < 0.001$), formed by all other models. Within this second group, Resnik is numerically the best model and shows a significant correlation with human data; nevertheless, the difference to the first group is evident from its substantially lower correlation coefficient.

The construction of the Pado dataset allows a further analysis. As mentioned in Section 4, the dataset consists of verb-argument pairs drawn from two different corpora. Therefore, each verb is combined both with some arguments that are seen in FrameNet, and some that are not. Our hypothesis is that the FrameNet-trained Pado model performs considerably better on the 216 "FN-Seen" data points (verb-argument pairs observed in FrameNet in at least one relation) than on the 198 "FN-Unseen" data points (verb-argument pairs unseen in both relations).

Table 4 shows the results of this analysis for the best-performing models. We observe a pattern corresponding to our expectations: The performance of the Pado model is clearly worse for FN-Unseen than for FN-Seen, while the Resnik and Parsed Cosine models perform more evenly across both datasets. While the Pado model is significantly better on the FN-Seen dataset, it is numerically outperformed by the Parsed Cosine model for the FN-Unseen data points. We conclude that the deep model is more accurate within the coverage of its resources, but loses its advantage when it has to resort to smoothing.

Model combination. Our last analysis indicates that the models have complementary strengths: the thematic role-based Pado model is the best plausibility predictor on the data points it has seen, while the Parsed cosine model overall predicts human data only numerically worse, and with better coverage. We therefore suggest to combine the predictions of the two models to combine their respective strengths.

For the moment, we only consider a naive backoff scheme: For each data point, we use the prediction of the Pado model if the data point is "FN-Seen" (cf. the last paragraph), and the prediction of the Parsed Cosine model otherwise. Note that this criterion does not consider the predictions of the models themselves, only properties of the underlying training set.

The actual combination requires a normalisation of the respective predictions, since one of the models (Pado) is probabilistic, while the other one (Parsed Cosine) is similarity-based, and their predictions are not directly comparable. We perform a simple normalisation by z-transforming the complete predictions of each model.[3] The combination of the scaled predictions in fact results in an improved correlation with the human data. The correlation coefficient of ρ=0.552 numerically exceeds either base model, and the coverage of 98% corresponds to the coverage of the more robust Parsed Cosine model.

We take this result as evidence that even a simple combination technique can lead to improved predictions. Unfortunately, our naive backoff scheme does not directly carry over to the McRae dataset, where only 2 out of 100 data points are "FN-Seen", and the Pado model would thus hardly contribute.

[3] The z transformation scales a dataset to a mean of 0 and a standard deviation of 1.

Model	FN-Seen Data		FN-Unseen Data	
Parsed Cosine	94%	0.426, ***	100%	0.461, ***
Resnik	96%	0.217, **	100%	0.263, ***
Pado	97%	0.569, ***	96%	0.383, ***

Table 4: Performance on data points seen and unseen in FrameNet (Pado dataset). **: $p < 0.01$ ***: $p < 0.001$

Discussion. We have verified experimentally that our vector similarity model is able to match the performance of a deep plausibility model, exceeding it in coverage, and to outperform a WordNet-based selectional preference model. We conclude that a completely corpus-driven approach constitutes a viable alternative to resource-based models.

One insight from our experiments is that vector similarity models constructed from dependency-parsed corpora perform significantly better than unparsed models. This indicates that dependency relations like `subject` and `object` are reliable syntactic correlates of semantic relations like *agent* and *patient*, but that their approximation in terms of word order introduces considerable noise. The Parsed models are best combined with Cosine Distance. We surmise that Cosine, which tends to consider low-frequency words more than Jaccard, is more susceptible to the additional noise in unparsed corpora.

Furthermore, the choice of basis elements for the vector space is vital: Plausibilities could only be predicted successfully with word-relation pairs as basis elements. This is in contrast to recent results on predominant sense acquisition, the task of identifying the most frequent sense for a given word in an unsupervised manner (McCarthy et al., 2004). On that task, Padó and Lapata (2007) found vector spaces with words as basis elements are in fact competitive with models using word-relation pairs. This divergence underlines an interesting difference between the two tasks. Evidently, predominant senses identification, as a WSD-related task, can succeed on the basis of topical information, which is represented well in word-based spaces. In contrast, plausibility judgments can only be predicted by a space based on word-relation pairs which can represent the finer-grained distinctions arising from different *relations* between verb and noun.

A second important finding is that the relative performance of the different models is the same on the McRae and Pado datasets. The Pado model performs best, followed by our Parsed Cosine vector similarity model, followed by the Unparsed and Resnik models.

The McRae dataset, however, is much more difficult to account for than the Pado data, independent of the model. This effect was already noted by Padó et al. (2006), who attributed it to the very limited overlap between the McRae dataset and FrameNet. While this explanation can account for the difference for the Pado model, we observe the same pattern across all models. This suggests that a more general frequency effect is at work here: The median frequency of the hand-selected McRae nouns is 1,356 in the BNC, as opposed to 8,184 for the corpus-derived Pado nouns. The resulting sparseness affects all model families, since all ultimately rely on co-occurrences.

The performance difference between the two datasets is particularly large for the WordNet-based selectional preference model (Resnik). A further analysis of the model's predictions shows that the model has difficulty in distinguishing between verb-relation-argument triples that differ only in the argument, such as $(shoot, agent, hunter)$ and $(shoot, agent, deer)$. Recall that it is crucial for the prediction of the McRae data to make this distinction, since the arguments for each relation are chosen to differ widely in plausibility. The reason for the Resnik model's difficulty is that arguments are mapped onto WordNet synsets, and whenever two arguments are mapped onto closely related synsets, their plausibility ratings are similar. This problem is graver for the McRae test set, where all arguments are animates, and thus more similar in terms of WordNet, than for the Pado set, which also contains a portion of inanimate arguments with animate counterparts. This analysis highlights again the fundamental problem of resource-based models, where design decisions of the underlying resource may limit, or even mislead, the models' generalisations.

Finally, we have shown in a first experiment that the syntax-based vector similarity model can be combined with the role-base model to obtain a combined model that performs superior to both. In this combined model, the shallow model's better coverage supplements the accurate predictions of the deep model.

6 Conclusions

In this paper, we have considered the computational modelling of human plausibility judgements for verb-relation-argument triples, a task equivalent to the computation of selectional preferences. We have extended a recent proposal (Erk, 2007) which combines ideas from selectional preference induction and vector space models. Our model can be constructed from a large corpus with partial syntactic information (specifically, subject and object relations) from which it builds an optimally informative vector space.

We have demonstrated that the successful evaluation of the model in Erk (2007) on the coarse-grained pseudo-word disambiguation task carries over to the prediction of human plausibility judgments which requires relatively fine-grained, relation-based distinctions. Our model is competitive with existing "deep" models while exhibiting a higher coverage. We have also shown that our vector similarity model can be combined with a "deep" model so that the combined model outperforms both base models. A thorough investigation of strategies for prediction combination and scaling remains future work.

The strategy of our model to derive generalisations directly from corpus data, without recourse to resources, is similar to another family of corpus-driven selectional preference models, namely EM-based clustering models (Rooth et al., 1999). However, we believe that our model has a number of advantages. (1), It is conceptually simple and implements the intuition behind selectional preference models, "generalise from known headwords to unknown ones", particularly directly through the comparison of new headwords to known ones according to a given definition of similarity. (2), The separation of the similarity computation and the acquisition of seen headwords gives the experimenter fine-grained control over the types and sources of information which inform the construction of the model. (3), The instantiation of the similarity computation with a vector space makes it possible to integrate additional linguistic information beyond verb-argument co-occurrences into the model, building on a large body of work in vector space construction. In sum, our modular model provides a higher degree of control than one-step models like the EM-based proposal.

An important avenue of further research is the ability of the vector plausibility model to model finer-grained distinctions between semantic relations beyond the agent/patient dichotomy, as thematic role-based models are able to. Excluding the direct use of role-annotated corpora like FrameNet for coverage reasons, the most promising strategy is to extend our present scheme of approximating semantic relations by grammatical realisations. How much noise this approximation introduces when finer role sets are used is an open research question.

Acknowledgments. The work presented in this paper was supported by the financial support of DFG (grants Pi-154/9-2 and IRTG "Language Technology and Cognitive Systems").

References

Naoki Abe and Hang Li. 1996. Learning word association norms using tree cut pair models. In *Proceedings of ICML 1996*, pages 3–11.

Lou Burnard, 1995. *User's guide for the British National Corpus*. British National Corpus Consortium, Oxford University Computing Services.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th ACL*, Prague, Czech Republic.

Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42th ACL*, pages 478–485, Barcelona, Spain.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th ACL*, pages 25–32, College Park, MA.

Dekang Lin. 1993. Principle-based parsing without overgeneration. In *Proceedings of the 31st ACL*, pages 112–120, Columbus, OH.

Will Lowe and Scott McDonald. 2000. The direct route: Mediated priming in semantic space. In *Proceedings of the 22nd CogSci*, pages 675–680, Philadelphia, PA.

Will Lowe. 2001. Towards a theory of semantic space. In *Proceedings of the 23rd CogSci*, pages 576–581, Edinburgh, UK.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs and adjectives using automatically acquired selectional preferences. *Computatinal Linguistics*, 29(4):639–654.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42th ACL*, pages 279–286, Barcelona, Spain.

Scott McDonald and Chris Brew. 2004. A distributional model of semantic context effects in lexical processing. In *Proceedings of the 42th ACL*, pages 17–24, Barcelona, Spain.

Scott McDonald and Will Lowe. 1998. Modelling functional priming and the associative boost. In *Proceedings of the 20th CogSci*, pages 675–680, Madison, WI.

Ken McRae, Michael Spivey-Knowlton, and Michael Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38:283–312.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2).

Ulrike Padó, Frank Keller, and Matthew W. Crocker. 2006. Combining syntax and thematic fit in a probabilistic model of sentence processing. In *Proceedings of the 28th CogSci*, pages 657–662, Vancouver, BC.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL 2007*, Rochester, NY.

Trivellore Raghunathan. 2003. An approximate test for homogeneity of correlated correlations. *Quality and Quantity*, 37:99–110.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing an semantically annotated lexicon via EM-based clustering. In *Proceedings of the 37th ACL*, pages 104–111, College Park, MA.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector-space model for information retrieval. *Journal of the American Society for Information Science*, 18:613–620.

John Trueswell, Michael Tanenhaus, and Susan Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language*, 33:285–318.

V-Measure: A conditional entropy-based external cluster evaluation measure

Andrew Rosenberg and Julia Hirschberg
Department of Computer Science
Columbia University
New York, NY 10027
{amaxwell,julia}@cs.columbia.edu

Abstract

We present V-measure, an external entropy-based cluster evaluation measure. V-measure provides an elegant solution to many problems that affect previously defined cluster evaluation measures including 1) dependence on clustering algorithm or data set, 2) the "problem of matching", where the clustering of only a portion of data points are evaluated and 3) accurate evaluation and combination of two desirable aspects of clustering, homogeneity and completeness. We compare V-measure to a number of popular cluster evaluation measures and demonstrate that it satisfies several desirable properties of clustering solutions, using simulated clustering results. Finally, we use V-measure to evaluate two clustering tasks: document clustering and pitch accent type clustering.

1 Introduction

Clustering techniques have been used successfully for many natural language processing tasks, such as document clustering (Willett, 1988; Zamir and Etzioni, 1998; Cutting et al., 1992; Vempala and Wang, 2005), word sense disambiguation (Shin and Choi, 2004), semantic role labeling (Baldewein et al., 2004), pitch accent type disambiguation (Levow, 2006). They are particularly appealing for tasks in which there is an abundance of language data available, but manual annotation of this data is very resource-intensive. Unsupervised clustering can eliminate the need for (full) manual annotation of the data into desired classes, but often at the cost of making evaluation of success more difficult.

External evaluation measures for clustering can be applied when class labels for each data point in some evaluation set can be determined *a priori*. The clustering task is then to assign these data points to any number of clusters such that each cluster contains all and only those data points that are members of the same class Given the ground truth class labels, it is trivial to determine whether this perfect clustering has been achieved. However, evaluating how far from perfect an incorrect clustering solution is a more difficult task (Oakes, 1998) and proposed approaches often lack rigor (Meila, 2007).

In this paper, we describe a new entropy-based external cluster evaluation measure, V-MEASURE[1], designed to address the problem of quantifying such imperfection. Like all external measures, V-measure compares a target clustering — e.g., a manually annotated representative subset of the available data — against an automatically generated clustering to determine now similar the two are. We introduce two complementary concepts, completeness and homogeneity, to capture desirable properties in clustering tasks.

In Section 2, we describe V-measure and how it is calculated in terms of homogeneity and completeness. We describe several popular external cluster evaluation measures and draw some comparisons to V-measure in Section 3. In Section 4, we discuss how some desirable properties for clustering are satisfied by V-measure vs. other measures. In Section 5, we present two applications of V-measure, on document clustering and on pitch accent type clustering.

2 V-Measure and Its Calculation

V-measure is an entropy-based measure which explicitly measures how successfully the criteria of homogeneity and completeness have been satisfied. V-measure is computed as the harmonic mean of distinct homogeneity and completeness scores, just as

[1] The 'V' stands for "validity", a common term used to describe the goodness of a clustering solution.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 410–420, Prague, June 2007. ©2007 Association for Computational Linguistics

precision and recall are commonly combined into F-measure (Van Rijsbergen, 1979). As F-measure scores can be weighted, V-measure can be weighted to favor the contributions of homogeneity or completeness.

For the purposes of the following discussion, assume a data set comprising N data points, and two partitions of these: a set of classes, $C = \{c_i | i = 1, \ldots, n\}$ and a set of clusters, $K = \{k_i | 1, \ldots, m\}$. Let A be the contingency table produced by the clustering algorithm representing the clustering solution, such that $A = \{a_{ij}\}$ where a_{ij} is the number of data points that are members of class c_i and elements of cluster k_j.

To discuss cluster evaluation measures we introduce two criteria for a clustering solution: homogeneity and completeness. A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class. A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster. The homogenity and completeness of a clustering solution run roughly in opposition: Increasing the homogeneity of a clustering solution often results in decreasing its completeness. Consider, two degenerate clustering solutions. In one, assigning every datapoint into a single cluster, guarantees perfect completeness — all of the data points that are members of the same class are trivially elements of the same cluster. However, this cluster is as *un*homogeneous as possible, since all classes are included in this single cluster. In another solution, assigning each data point to a distinct cluster guarantees perfect homogeneity — each cluster trivially contains only members of a single class. However, in terms of completeness, this solution scores very poorly, unless indeed each class contains only a single member. We define the distance from a perfect clustering is measured as the weighted harmonic mean of measures of homogeneity and completeness.

Homogeneity:

In order to satisfy our homogeneity criteria, a clustering must assign **only** those datapoints that are members of a single class to a single cluster. That is, the class distribution within each cluster should be skewed to a single class, that is, zero entropy. We determine how close a given clustering is to this ideal by examining the conditional entropy of the class distribution given the proposed clustering. In the perfectly homogeneous case, this value, $H(C|K)$, is 0. However, in an imperfect situation, the size of this value, in bits, is dependent on the size of the dataset and the distribution of class sizes. Therefore, instead of taking the raw conditional entropy, we normalize this value by the maximum reduction in entropy the clustering information could provide, specifically, $H(C)$.

Note that $H(C|K)$ is maximal (and equals $H(C)$) when the clustering provides no new information — the class distribution within each cluster is equal to the overall class distribiution. $H(C|K)$ is 0 when each cluster contains only members of a single class, a perfectly homogenous clustering. In the degenerate case where $H(C) = 0$, when there is only a single class, we define homogeneity to be 1. For a perfectly homogenous solution, this normalization, $\frac{H(C|K)}{H(C)}$, equals 0. Thus, to adhere to the convention of 1 being desirable and 0 undesirable, we define homogeneity as:

$$h = \begin{cases} 1 & \text{if } H(C,K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{else} \end{cases} \tag{1}$$

where

$$\begin{aligned} H(C|K) &= -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \\ H(C) &= -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \end{aligned}$$

Completeness:

Completeness is symmetrical to homogeneity. In order to satisfy the completeness criteria, a clustering must assign **all** of those datapoints that are members of a single class to a single cluster. To evaluate completeness, we examine the distribution of cluster assignments within each class. In a perfectly complete clustering solution, each of these distributions will be completely skewed to a single cluster. We can evaluate this degree of skew by calculating the conditional entropy of the proposed cluster distribution given the class of the component datapoints, $H(K|C)$. In the perfectly complete case, $H(K|C) = 0$. However, in the worst case scenario, each class is represented by every cluster with a distribution equal to the distribution of cluster sizes, $H(K|C)$ is maximal and equals $H(K)$. Finally, in the degenerate case where $H(K) = 0$, when there is a single cluster, we define completeness to be 1. Therefore, symmetric to the calculation above, we define completeness as:

$$c = \begin{cases} 1 & \text{if } H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{else} \end{cases} \tag{2}$$

where

$$\begin{aligned} H(K|C) &= -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{k=1}^{|K|} a_{ck}} \\ H(K) &= -\sum_{k=1}^{|K|} \frac{\sum_{c=1}^{|C|} a_{ck}}{n} \log \frac{\sum_{c=1}^{|C|} a_{ck}}{n} \end{aligned}$$

Based upon these calculations of homogeneity and completeness, we then calculate a clustering solution's V-measure by computing the weighted harmonic mean of homogeneity and completeness, $V_\beta = \frac{(1+\beta)*h*c}{(\beta*h)+c}$. Similarly to the familiar F-measure, if β is greater than 1 completeness is weighted more strongly in the calculation, if β is less than 1, homogeneity is weighted more strongly.

Notice that the computations of homogeneity, completeness and V-measure are completely independent of the number of classes, the number of clusters, the size of the data set and the clustering algorithm used. Thus these measures can be applied to and compared across any clustering solution, regardless of the number of data points (n-invariance), the number of classes or the number of clusters. Moreover, by calculating homogeneity and completeness separately, a more precise evaluation of the performance of the clustering can be obtained.

3 Existing Evaluation Measures

Clustering algorithms divide an input data set into a number of partitions, or clusters. For tasks where some target partition can be defined for testing purposes, we define a "clustering solution" as a mapping from each data point to its cluster assignments in both the target and hypothesized clustering. In the context of this discussion, we will refer to the target partitions, or clusters, as CLASSES, referring only to hypothesized clusters as CLUSTERS.

Two commonly used external measures for assessing clustering success are $Purity$ and $Entropy$ (Zhao and Karypis, 2001), defined as,

$$Purity = \sum_{r=1}^{k} \frac{1}{n} \max_i(n_r^i)$$

$$Entropy = \sum_{r=1}^{k} \frac{n_r}{n}\left(-\frac{1}{\log q} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r}\right)$$

where q is the number of classes, k the number of clusters, n_r is the size of cluster r, and n_r^i is the number of data points in class i clustered in cluster r.

Both these approaches represent plausable ways to evaluate the homogeneity of a clustering solution. However, our completeness criterion is not measured at all. That is, they do not address the question of whether all members of a given class are included in a single cluster. Therefore the $Purity$ and $Entropy$ measures are likely to improve (increased $Purity$, decreased $Entropy$) monotonically with the number of clusters in the result, up to a degenerate maximum where there are as many clusters as data points. However, clustering solutions rated high by either measure may still be far from ideal.

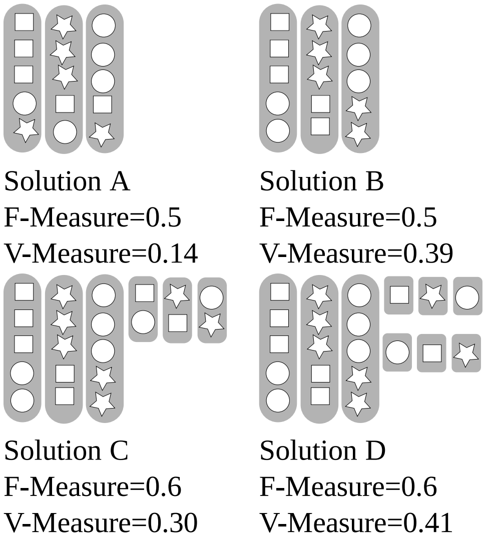
Another frequently used external clustering evaluation measure is commonly refered to as "clustering accuracy". The calculation of this accuracy is inspired by the information retrieval metric of F-Measure (Van Rijsbergen, 1979). The formula for this clustering F-measure as described in (Fung et al., 2003) is shown in Figure 3.

Let N be the number of data points, C the set of classes, K the set of clusters and n_{ij} be the number of members of class $c_i \in C$ that are elements of cluster $k_j \in K$.

$$\begin{aligned} F(C,K) &= \sum_{c_i \in C} \frac{|c_i|}{N} \max_{k_j \in K}\{F(c_i,k_j)\} \qquad (3) \\ F(c_i,k_j) &= \frac{2 * R(c_i,k_j) * P(c_i,k_j)}{R(c_i,k_j) + P(c_i,k_j)} \\ R(c_i,k_j) &= \frac{n_{ij}}{|c_i|} \\ P(c_i,k_j) &= \frac{n_{ij}}{|k_j|} \end{aligned}$$

Figure 1: Calculation of clustering F-measure

This measure has a significant advantage over $Purity$ and $Entropy$, in that it does measure both the homogeneity and the completeness of a clustering solution. Recall is calculated as the portion of items from class i that are present in cluster j, thus measuring how complete cluster j is with respect to class i. Similarly, Precision is calculated as the por-



Solution A
F-Measure=0.5
V-Measure=0.14

Solution B
F-Measure=0.5
V-Measure=0.39

Solution C
F-Measure=0.6
V-Measure=0.30

Solution D
F-Measure=0.6
V-Measure=0.41

Figure 2: Examples of the Problem of Matching

tion of cluster j that is a member of class i, thus measuring how homogenous cluster j is with respect to class i.

Like some other external cluster evaluation techniques (misclassification index (MI) (Zeng et al., 2002), H (Meila and Heckerman, 2001), L (Larsen and Aone, 1999), D (van Dongen, 2000), micro-averaged precision and recall (Dhillon et al., 2003)), F-measure relies on a post-processing step in which each cluster is assigned to a class. These techniques share certain problems. First, they calculate the goodness not only of the given clustering solution, but also of the cluster-class matching. Therefore, in order for the goodness of two clustering solutions to be compared using one these measures, an identical post-processing algorithm must be used. This problem can be trivially addressed by fixing the class-cluster matching function and including it in the definition of the measure as in H. However, a second and more critical problem is the "problem of matching" (Meila, 2007). In calculating the similarity between a hypothesized clustering and a 'true' clustering, these measures only consider the contributions from those clusters that are matched to a target class. This is a major problem, as two significantly different clusterings can result in identical scores.

In figure 2, we present some illustrative examples of the problem of matching. For the purposes of this discussion we will be using F-Measure as the measure to describe the problem of matching, however, these problems affect any measure which requires a mapping from clusters to classes for evaluation.

In the figures, the shaded regions represent CLUSTERS, the shapes represent CLASSES. In a perfect clustering, each shaded region would contain all and only the same shapes. The problem of matching can manifest itself either by not evaluating the entire membership of a cluster, or by not evaluating every cluster. The former situation is presented in the figures A and B in figure 2. The F-Measure of both of these clustering solutions in 0.6. (The precision and recall for each class is $\frac{3}{5}$.) That is, for each class, the best or "matched" cluster contains 3 of 5 elements of the class (Recall) and 3 of 5 elements of the cluster are members of the class (Precision). The make up of the clusters beyond the majority class is not evaluated by F-Measure. Solution B is a better clustering solution than solution A, in terms of both homogeneity (crudely, "each cluster contains fewer[2] classes") and completeness ("each class is contained in fewer clusters"). Indeed, the V-Measure of solution B (0.387) is greater than that of solution A (0.135). Solutions C and D represent a case in which not every cluster is considered in the evaluation of F-Measure. In this example, the F-Measure of both solutions is 0.5 (the harmonic mean of $\frac{3}{5}$ and $\frac{3}{7}$). The small "unmatched" clusters are not measured at all in the calculation of F-Measure. Solution D is a better clustering than solution C – there are no incorrect clusterings of different classes in the small clusters. V-Measure reflects this, solution C has a V-measure of 0.30 while the V-measure of solution D is 0.41.

A second class of clustering evaluation techniques is based on a combinatorial approach which examines the number of pairs of data points that are clustered similarly in the target and hypothesized clustering. That is, each pair of points can either be 1) clustered together in both clusterings (N_{11}), 2) clustered separately in both clusterings (N_{00}), 3) clustered together in the hypothesized but not the target clustering (N_{01}) or 4) clustered together in the target but not in the hypothesized clustering (N_{10}). Based on these 4 values, a number of measures have been proposed, including Rand Index (Rand, 1971),

[2] Homogeneity is not measured by V-measure as a count of the number of classes contained by a cluster but "fewer" is an acceptable way to conceptualize this criterion for the purposes of these examples.

Adjusted Rand Index (Hubert and Arabie, 1985), Γ statistic (Hubert and Schultz, 1976), Jaccard (Milligan et al., 1983), Fowlkes-Mallows (Fowlkes and Mallows, 1983) and Mirkin (Mirkin, 1996). We illustrate this class of measures with the calculation of Rand Index. $Rand(C,K) = \frac{N_{11}+N_{00}}{n(n-1)/2}$ Rand Index can be interpreted as the probability that a pair of points is clustered similarly (together or separately) in C and K.

Meila (2007) describes a number of potential problems of this class of measures posed by (Fowlkes and Mallows, 1983) and (Wallace, 1983). The most basic is that these measures tend not to vary over the interval of $[0,1]$. Transformations like those applied by the adjusted Rand Index and a minor adjustment to the Mirkin measure (see Section 4) can address this problem. However, pair matching measures also suffer from distributional problems. The baseline for Fowlkes-Mallows varies significantly between 0.6 and 0 when the ratio of data points to clusters is greater than 3 — thus including nearly all real-world clustering problems. Similarly, the Adjusted Rand Index, as demonstrated using Monte Carlo simulations in (Fowlkes and Mallows, 1983), varies from 0.5 to 0.95. This variance in the measure's baseline prompts Meila to ask if the assumption of linearity following normalization can be maintained. If the behavior of the measure is so unstable before normalization can users reasonably expect stable behavior **following** normalization?

A final class of cluster evaluation measures are based on information theory. These measures analyze the distribution of class and cluster membership in order to determine how successful a given clustering solution is or how different two partitions of a data set are. We have already examined one member of this class of measures, $Entropy$. From a coding theory perspective, $Entropy$ is the weighted average of the code lengths of each cluster. Our V-measure is a member of this class of clustering measures. One significant advantage that information theoretic evaluation measures have is that they provide an elegant solution to the "problem of matching". By examining the relative sizes of the classes and clusters being evaluated, these measures all evaluate the entire membership of each cluster — not just a 'matched' portion.

Dom's Q_0 measure (Dom, 2001) uses conditional entropy, $H(C|K)$ to calculate the goodness of a clustering solution. That is, given the hypothesized partition, what is the number of bits necessary to represent the true clustering?

However, this term – like the $Purity$ and $Entropy$ measures – only evaluates the homogeneity of a solution. To measure the completeness of the hypothesized clustering, Dom includes a model cost term calculated using a coding theory argument. The overall clustering quality measure presented is the sum of the costs of representing the data ($H(C|K)$) and the model. The motivation for this approach is an appeal to parsimony: Given identical conditional entropies, $H(C|K)$, the clustering solution with the fewest clusters should be preferred. Dom also presents a normalized version of this term, Q_2, which has a range of $(0,1]$ with greater scores being representing more preferred clusterings.

$$Q_0(C,K) = H(C|K) + \frac{1}{n}\sum_{k=1}^{|K|} \log \binom{h(k)+|C|-1}{|C|-1}$$

where C is the target partition, K is the hypothesized partition and $h(k)$ is the size of cluster k.

$$Q_2(C,K) = \frac{\frac{1}{n}\sum_{c=1}^{|C|} \log \binom{h(c)+|C|-1}{|C|-1}}{Q_0(C,K)}$$

We believe that V-measure provides two significant advantages over Q_0 that make it a more useful diagnostic tool. First, Q_0 does not explicitly calculate the degree of completeness of the clustering solution. The cost term captures some of this information, since a partition with fewer clusters is likely to be more complete than a clustering solution with more clusters. However, Q_0 does not explicitly address the interaction between the conditional entropy and the cost of representing the model. While this is an application of the *minimum description length* (MDL) principle (Rissanen, 1978; Rissanen, 1989), it does not provide an intuitive manner for assessing our two competing criteria of homogeneity and completeness. That is, at what point does an increase in conditional entropy (homogeneity) justify a reduction in the number of clusters (completeness).

Another information-based clustering measure is variation of information (VI) (Meila, 2007), $VI(C,K) = H(C|K) + H(K|C)$. VI is presented as a distance measure for comparing partitions (or clusterings) of the same data. It therefore does not distinguish between hypothesized and target clusterings. VI has a number of useful properties. First, it satisfies the metric axioms. This quality allows users to intuitively understand how VI values combine and relate to one another. Secondly, it is "convexly additive". That is to say, if a cluster is split, the distance from the new cluster to the original is the distance induced by the split times the size of the cluster. This property guarantees that all changes to the metric are "local": the impact of splitting or merging clusters is limited to only those clusters involved, and its size is relative to the size of these clusters. Third, VI is n-invariant: the number of data points in the cluster do not affect the value of the measure. VI depends on the relative sizes of the partitions of C and K, not on the number of points in these partitions. However, VI is bounded by the maximum number of clusters in C or K, k^*. Without manual modification however, $k^* = n$, where each cluster contains only a single data point. Thus, while technically n-invariant, the possible values of VI are heavily dependent on the number of data points being clustered. Thus, it is difficult to compare VI values across data sets and clustering algorithms without fixing k^*, as VI will vary over different ranges. It is a trivial modification to modify VI such that it varies over [0,1]. Normalizing, VI by $\log n$ or $1/2 \log k^*$ guarantee this range. However, Meila (2007) raises two potential problems with this modification. The normalization should not be applied if data sets of different sizes are to be compared — it negates the n-invariance of the measure. Additionally, if two authors apply the latter normalization and do not use the same value for k^*, their results will not be comparable.

While VI has a number of very useful distance properties when analyzing a single data set across a number of settings, it has limited utility as a general purpose clustering evaluation metric for use across disparate clusterings of disparate data sets. Our homogeneity (h) and completeness (c) terms both range over [0,1] and are completely n-invariant and k^*-invariant. Furthermore, measuring each as a ratio of bit lengths has greater intuitive appeal than a more opportunistic normalization.

V-measure has another advantage as a clustering evaluation measure over VI and Q_0. By evaluating homogeneity and completeness in a symmetrical, complementary manner, the calculation of V-measure makes their relationship clearly observable. Separate analyses of homogeneity and completeness are not possible with any other cluster evaluation measure. Moreover, by using the harmonic mean to combine homogeneity and completeness, V-measure is unique in that it can also prioritize one criterion over another, depending on the clustering task and goals.

4 Comparing Evaluation Measures

Dom (2001) describes a parametric technique for generating example clustering solutions. He then proceeds to define five "desirable properties" that clustering accuracy measures should display, based on the parameters used to generate the clustering solution. To compare V-measure more directly to alternative clustering measures, we evaluate V-measure and other measures against these and two additional desirable properties.

The parameters used in generating a clustering solution are as follows.

- $|C|$ The number of classes
- $|K|$ The number of clusters
- $|K_{noise}|$ Number of "noise" clusters; $|K_{noise}| < |K|$
- $|C_{noise}|$ Number of "noise" classes; $|C_{noise}| < |C|$
- ϵ Error probability; $\epsilon = \epsilon_1 + \epsilon_2 + \epsilon_3$.
- ϵ_1 The error mass within "useful" class-cluster pairs
- ϵ_2 The error mass within noise clusters
- ϵ_3 The error mass within noise classes

The construction of a clustering solution begins with a matching of "useful" clusters to "useful" classes[3]. There are $|K_u| = |K| - |K_{noise}|$ "useful" clusters and $|C_u| = |C| - |C_{noise}|$ "useful" classes. The claim is useful classes and clusters are matched to each other and matched pairs contain more data points than unmatched pairs. Probability mass of $1 - \epsilon$ is evenly distributed across each match. Error mass of ϵ_1 is evenly distributed across each pair

[3] The operation of this matching is omitted in the interest of space. Interested readers should see (Dom, 2001).

of non-matching useful class/cluster pairs. Noise clusters are those that contain data points equally from each cluster. Error mass of ϵ_2 is distributed across every "noise"-cluster/ "useful"-class pair. We extend the parameterization technique described in (Dom, 2001) in with $|C_{noise}|$ and ϵ_3. Noise classes are those that contain data points equally from each cluster. Error mass of ϵ_3 is distributed across every "useful"-cluster/"noise"-class pair. An example solution, along with its generating parameters is given in Figure 3.

	C_1	C_2	C_3	C_{noise1}
K_1	12	12	2	3
K_2	2	2	12	3
K_{noise1}	4	4	4	0

Figure 3: Sample parametric clustering solution with $n = 60, |K| = 3, |K_{noise}| = 1, |C| = 3, |C_{noise}| = 1, \epsilon_1 = .1, \epsilon_2 = .2, \epsilon_3 = .1$

The desirable properties proposed by Dom are given as P1-P5 in Table 1. We include two additional properties (P6,P7) relating the examined measure value to the number of 'noise' classes and ϵ_3.

P1 For $|K_u| < |C|$ and $\Delta|K_u| \leq (|C| - |K_u|)$, $\frac{\Delta M}{\Delta|K_u|} > 0$

P2 For $|K_u| \geq |C|$, $\frac{\Delta M}{\Delta|K_u|} < 0$

P3 $\frac{\Delta M}{\Delta|K_{noise}|} < 0$, if $\epsilon_2 > 0$

P4 $\frac{\delta M}{\delta\epsilon_1} \leq 0$, with equality only if $|K_u| = 1$

P5 $\frac{\delta M}{\delta\epsilon_2} \leq 0$, with equality only if $|K_{noise}| = 0$

P6 $\frac{\Delta M}{\Delta|C_{noise}|} < 0$, if $\epsilon_3 > 0$

P7 $\frac{\delta M}{\delta\epsilon_3} \leq 0$, with equality only if $|C_{noise}| = 0$

Table 1: Desirable Properties of a cluster evaluation measure M

To evaluate how different clustering measures satisfy each of these properties, we systematically varied each parameter, keeping $|C| = 5$ fixed.

- $|K_u|$: 10 values: 2, 3,. . . , 11
- $|K_{noise}|$: 7 values: 0, 1,. . . , 6
- $|C_{noise}|$: 7 values: 0, 1,. . . , 6
- ϵ_1: 4 values: 0, 0.033, 0.066, 0.1
- ϵ_2: 4 values: 0, 0.066, 0.133, 0.2
- ϵ_3: 4 values: 0, 0.066, 0.133, 0.2

We evaluated the behavior of V-Measure, Rand, Mirkin, Fowlkes-Mallows, Gamma, Jaccard, VI, Q_0, F-Measure against the desirable properties P1-P7[4]. Based on the described systematic modification of each parameter, only V-measure, VI and Q_0 empirically satisfy all of P1-P7 in all experimental conditions. Full results reporting how frequently each evaluated measure satisfied the properties based on these experiments can be found in table 2.

All evaluated measures satisfy P4 and P7. However, Rand, Mirkin, Fowlkes-Mallows, Gamma, Jaccard and F-Measure all fail to satisfy P3 and P6 in at least one experimental configuration. This indicates that the number of 'noise' classes or clusters can be increased without reducing any of these measures. This implies a computational obliviousness to potentially significant aspects of an evaluated clustering solution.

5 Applying V-measure

In this section, we present two clustering experiments. We describe a document clustering experiment and evaluate its results using V-measure, highlighting the interaction between homogeneity and completeness. Second, we present a pitch accent type clustering experiment. We present results from both of these experiments in order to show how V-measure can be used to drawn comparisons across data sets.

5.1 Document Clustering

Clustering techniques have been used widely to sort documents into topic clusters. We reproduce such an experiment here to demonstrate the usefulness of V-measure. Using a subset of the TDT-4 corpus (Strassel and Glenn, 2003) (1884 English news wire and broadcast news documents manually labeled with one of 12 topics), we ran clustering experiments using k-means clustering (McQueen, 1967) and evaluated the results using V-Measure, VI and Q_0 – those measures that satisfied the desirable properties defined in section 4. The topics and relative distributions are as follows: Acts

[4] The inequalities in the desirable properties are inverted in the evaluation of VI, Q_0 and Mirkin as they are defined as distance, as opposed to similarity, measures.

Property	Rand	Mirkin	Fowlkes	Γ	Jaccard	F-measure	Q0	VI	V-Measure
P1	0.18	0.22	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P2	1.0	1.0	0.76	1.0	0.89	0.98	1.0	1.0	1.0
P3	0.0	0.0	0.30	0.19	0.21	0.0	1.0	1.0	1.0
P4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P5	0.50	0.57	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P6	0.20	0.20	0.41	0.26	0.52	0.87	1.0	1.0	1.0
P7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 2: Rates of satisfaction of desirable properties

of Violence/War (22.3%), Elections (14.4%), Diplomatic Meetings (12.9%), Accidents (8.75%), Natural Disasters (7.4%), Human Interest (6.7%), Scandals (6.5%), Legal Cases (6.4%), Miscellaneous (5.3%), Sports (4.7), New Laws (3.2%), Science and Discovery (1.4%).

We employed stemmed (Porter, 1980), tf*idf-weighted term vectors extracted for each document as the clustering space for these experiments, which yielded a very high dimension space. To reduce this dimensionality, we performed a simple feature selection procedure including in the feature vector only those terms that represented the highest tf*idf value for at least one data point. This resulted in a feature vector containing 484 tf*idf values for each document. Results from k-means clustering are are shown in Figure 4.

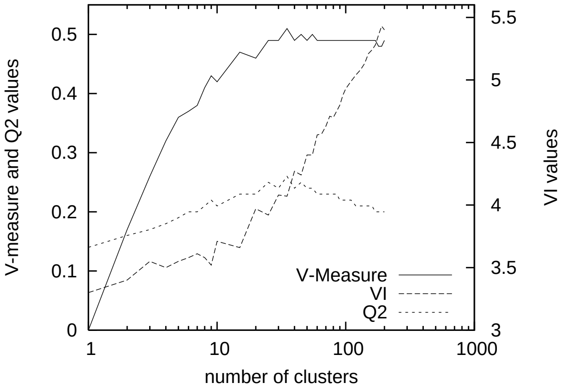


Figure 4: Results of document clustering measured by V-Measure, VI and Q_2

The first observation that can be drawn from these results is the degree to which VI is dependent on the number of clusters (k). This dependency severely limits the usefulness of VI: it is inappropriate in selecting an appropriate parameter for k or for evaluating the distance between clustering solutions generated using different values of k.

V-measure and Q_2 demonstrate similar behavior in evaluating these experimental results. They both reach a maximal value with 35 clusters, however, Q_2 shows a greater descent as the number of clusters increases. We will discuss this quality in greater detail in section 5.2.

5.2 Pitch Accent Clustering

Pitch accent is how speakers of many languages make a word intonational prominent. In most pitch accent languages, words can also be accented in different ways to convey different meanings (Hirschberg, 2002). In the ToBI labeling conventions for Standard American English (Silverman et al., 1992), for example, there are five different accent types (H*, L*, H+!H*, L+H*, L*+H).

We extracted a number of acoustic features from accented words within the read portion of the Boston Directions Corpus (BDC) (Nakatani et al., 1995) and examined how well clustering in these acoustic dimensions correlates to manually annotated pitch accent types. We obtained a very skewed distribution, with a majority of H* pitch accents.[5] We therefore included only a randomly selected 10% sample of H* accents, providing a more even distribution of pitch accent types for clustering: H* (54.4%), L*(32.1%), L+H* (26.5%), L*+H (2.8%), H+!H* (2.1%).

We extracted ten acoustic features from each accented word to serve as the clustering space for this experiment. Using Praat's (Boersma, 2001) Get Pitch (ac)... function, we calculated the mean F0 and ΔF0, as well as z-score speaker normalized versions of the same. We included in the feature vector the relative location of the maximum pitch value in the word as well as the distance between this max-

[5] Pitch accents containing a high tone may also be downstepped, or spoken in a compressed pitch range. Here we collapsed all DOWNSTEPPED instances of each pitch accent with the corresponding non-downstepped instances.

imum and the point of maximum intensity. Finally, we calculated the raw and speaker normalized slope from the start of the word to the maximum pitch, and from the maximum pitch to the end of the word.

Using this feature vector, we performed k-means clustering and evaluate how successfully these dimensions represent differences between pitch accent types. The resulting V-measure, VI and Q_0 calculations are shown in Figure 5.

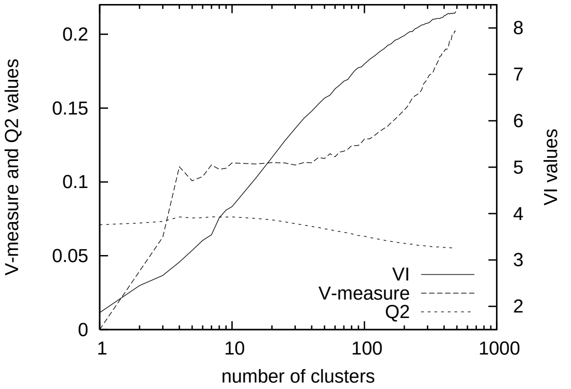


Figure 5: Results of pitch accent clustering measured by V-Measure, VI and Q_0

In evaluating the results from these experiments, Q_2 and V-measure reveal considerably different behaviors. Q_2 shows a maximum at $k = 10$, and descends at k increases. This is an artifact of the MDL principle. Q_2 makes the claim that a clustering solution based on fewer clusters is preferable to one using more clusters, and that the balance between the number of clusters and the conditional entropy, $H(C|K)$, should be measured in terms of coding length. With V-measure, we present a different argument. We contend that the a high value of k does not inherently reduce the goodness of a clustering solution. Using these results as an example, we find that at approximately 30 clusters an increase of clusters translates to an increase in V-Measure. This is due to an increased homogeneity ($\frac{H(C|K)}{H(C)}$) and a relatively stable completeness ($\frac{H(K|C)}{H(K)}$). That is, inclusion of more clusters leads to clusters with a more skewed within-cluster distribution and a equivalent distribution of cluster memberships within classes. This is intuitively preferable – one criterion is improved, the other is not reduced – despite requiring additional clusters. This is an instance in which the *MDL* principle limits the usefulness of Q_2. We again (see section 5.1) observe the close dependency of VI and k. Moreover, in considering figures 5 and 4, simultaneously, we see considerably higher values achieved by the document clustering experiments. Given the naïve approaches taken in these experiments, this is expected – and even desired – given the previous work on these tasks: document clustering has been notably more successfully applied than pitch accent clustering. These examples allow us to observe how transparently V-measure can be used to compare the behavior across distinct data sets.

6 Conclusion

We have presented a new external cluster evaluation measure, V-measure, and compared it with existing clustering evaluation measures. V-measure is based upon two criteria for clustering usefulness, homogeneity and completeness, which capture a clustering solution's success in including all and only datapoints from a given class in a given cluster. We have also demonstrated V-measure's usefulness in comparing clustering success across different domains by evaluating document and pitch accent clustering solutions. We believe that V-measure addresses some of the problems that affect other cluster measures. 1) It evaluates a clustering solution independent of the clustering algorithm, size of the data set, number of classes and number of clusters. 2) It does not require its user to map each cluster to a class. Therefore, it only evaluates the quality of the clustering, not a post-hoc class-cluster mapping. 3) It evaluates the clustering of every data point, avoiding the "problem of matching". 4) By evaluating the criteria of both homogeneity and completeness, V-measure is more comprehensive than those that evaluate only one. 5) Moreover, by evaluating these criteria separately and explicitly, V-measure can serve as an elegant diagnositic tool providing greater insight into clustering behavior.

Acknowledgments

The authors thank Kapil Thadani, Martin Jansche and Sasha Blair-Goldensohn and for their feedback. This work was funded in part by the DARPA GALE program under a subcontract to SRI International.

References

Ulrike Baldewein, Katrin Erk, Sebastian Pado, and Detlef Prescher. 2004. Semantic role labelling with similarity-based generalization using EM-based clustering. In *Proceedings of Senseval'04*, Barcelona.

Paul Boersma. 2001. Praat, a system for doing phonetics by computer. *Glot International*, 5(9-10):341–345.

Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. 1992. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329.

I. S. Dhillon, S. Mallela, and D. S. Modha. 2003. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 89–98.

Byron E. Dom. 2001. An information-theoretic external cluster-validity measure. Technical Report RJ10219, IBM, October.

E. B. Fowlkes and C. L. Mallows. 1983. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78:553–569.

Benjamin C. M. Fung, Ke Wang, and Martin Ester. 2003. Hierarchical document clustering using frequent itemsets. In *Proc. of the SIAM International Conference on Data Mining*.

Julia Hirschberg. 2002. The pragmatics of intonational meaning. In *Proc. Speech Prosody*, pages 65–68.

L. Hubert and P. Arabie. 1985. Comparing partitions. *Journal of Classification*, 2:193–218.

L. Hubert and J. Schultz. 1976. Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29:190–241.

Bjornar Larsen and Chinatsu Aone. 1999. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA. ACM Press.

Gina-Anne Levow. 2006. Unsupervised and semi-supervised learning of tone and pitch accent. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 224–231, Morristown, NJ, USA. Association for Computational Linguistics.

J. McQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proc. of the Fifty Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

Marina Meila and David Heckerman. 2001. An experimental comparison of model-based clustering methods. *Mach. Learn.*, 42(1/2):9–29.

Marina Meila. 2007. Comparing clusterings – an information based distance. *Journal of Multivariate Analysis*, 98:873–895.

G. W. Milligan, S. C. Soon, and L. M. Sokol. 1983. The effect of cluster size, dimensionality and the number of clustes on recovery of true cluster structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:40–47.

Boris G. Mirkin. 1996. *Mathematical classification and clustering*. Kluwer Academic Press.

Christine Nakatani, Julia Hirschberg, and Barbara Grosz. 1995. Discourse structure in spoken language: Studies on speech corpora. In *Working Notes of AAAI-95 Spring Symposiom on Empirical Methods in Discourse Interpretation*.

Michael P. Oakes. 1998. *Statistics for Corpus Linguistics*. Edinburgh University Press.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, Dec.

J. Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.

J. Rissanen. 1989. Stochastic complexity in statistical inquiry. *World Scientific Series in Computer Science*, 15.

Sa-Im Shin and Key-Sun Choi. 2004. Automatic word sense clustering using collocation for sense adaptation. In *The Second Global Wordnet Conference*.

K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. 1992. Tobi: A standard for labeling english prosody. In *Proc. of the 1992 International Conference on Spoken Language Processing*, volume 2, pages 12–16.

S. Strassel and M. Glenn. 2003. Creating the annotated tdt-4 y2003 evaluation corpus. http://www.nist.gov/speech/tests/tdt/tdt2003/papers/ldc.ppt.

Stijn van Dongen. 2000. Performance criteria for graph clustering and markov cluster experiments. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, The Netherlands.

C. J. Van Rijsbergen. 1979. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow.

Santosh Vempala and Grant Wang. 2005. The benefit of spectral projection for document clustering. In *Workshop on Clustering High Dimensional Data and its Applications Held in conjunction with Fifth SIAM International Conference on Data Mining (SDM 2005)*.

D. L. Wallace. 1983. Comment. *Journal of the American Statistical Association*, 78:569–576.

Peter Willett. 1988. Recent trends in hierarchic document clustering: a critical review. *Inf. Process. Manage.*, 24(5):577–597.

Oren Zamir and Oren Etzioni. 1998. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54.

Yujing Zeng, Jianshan Tang, Javier Garcia-Frias, and Guang R. Gao. 2002. An adaptive meta-clustering approach: Combining the information from different clustering results. *csb*, 00:276.

Ying Zhao and George Karypis. 2001. Criterion functions for ducument clustering: Experiments and analysis. Technical Report TR 01–40, Department of Computer Science, University of Minnesota.

Bayesian Document Generative Model with Explicit Multiple Topics

Issei Sato
Graduate School of Information Science
and Technology,
The University of Tokyo
sato@r.dl.itc.u-tokyo.ac.jp

Hiroshi Nakagawa
Information Technology Center,
The University of Tokyo
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract

In this paper, we proposed a novel probabilistic generative model to deal with explicit multiple-topic documents: Parametric Dirichlet Mixture Model(PDMM). PDMM is an expansion of an existing probabilistic generative model: Parametric Mixture Model(PMM) by hierarchical Bayes model. PMM models multiple-topic documents by mixing model parameters of each single topic with an equal mixture ratio. PDMM models multiple-topic documents by mixing model parameters of each single topic with mixture ratio following Dirichlet distribution. We evaluate PDMM and PMM by comparing F-measures using MEDLINE corpus. The evaluation showed that PDMM is more effective than PMM.

1 Introduction

Documents, such as those seen on Wikipedia and Folksonomy, have tended to be assigned with explicit multiple topics. In this situation, it is important to analyze a linguistic relationship between documents and the assigned multiple topics . We attempt to model this relationship with a probabilistic generative model. A probabilistic generative model for documents with multiple topics is a probability model of the process of generating documents with multiple topics. By focusing on modeling the generation process of documents and the assigned multiple topics, we can extract specific properties of documents and the assigned multiple topics. The model can also be applied to a wide range of applications such as automatic categorization for multiple topics, keyword extraction and measuring document similarity, for example.

A probabilistic generative model for documents with multiple topics is categorized into the following two models. One model assumes a topic as a latent topic. We call this model the latent-topic model. The other model assumes a topic as an explicit topic. We call this model the explicit-topic model.

In a latent-topic model, a latent topic indicates not a concrete topic but an underlying implicit topic of documents. Obviously this model uses an unsupervised learning algorithm. Representative examples of this kind of model are Latent Dirichlet Allocation(LDA)(D.M.Blei et al., 2001; D.M.Blei et al., 2003) and Hierarchical Dirichlet Process(HDP)(Y.W.Teh et al., 2003).

In an explicit-topic model, an explicit topic indicates a concrete topic such as economy or sports, for example. A learning algorithm for this model is a supervised learning algorithm. That is, an explicit topic model learns model parameter using a training data set of tuples such as (documents, topics). Representative examples of this model are Parametric Mixture Models(PMM1 and PMM2)(Ueda, N. and Saito, K., 2002a; Ueda, N. and Saito, K., 2002b). In the remainder of this paper, PMM indicates PMM1 because PMM1 is more effective than PMM2.

In this paper, we focus on the explicit topic model. In particular, we propose a novel model that is based on PMM but fundamentally improved.

The remaining part of this paper is organized as follows. Sections 2 explains terminology used in the

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 421–429, Prague, June 2007. ©2007 Association for Computational Linguistics

following sections. Section 3 explains PMM that is most directly related to our work. Section 4 points out the problem of PMM and introduces our new model. Section 5 evaluates our new model. Section 6 summarizes our work.

2 Terminology

This section explains terminology used in this paper. K is the number of explicit topics. V is the number of words in the vocabulary. $\mathscr{V} = \{1, 2, \cdots, V\}$ is a set of vocabulary index. $\mathscr{Y} = \{1, 2, \cdots, K\}$ is a set of topic index. N is the number of words in a document. $\boldsymbol{w} = (w_1, w_2, \cdots, w_N)$ is a sequence of N words where w_n denotes the nth word in the sequence. $\boldsymbol{w}$ is a document itself and is called words vector. $\boldsymbol{x} = (x_1, x_2, \cdots, x_V)$ is a word-frequency vector, that is, BOW(Bag Of Words) representation where x_v denotes the frequency of word v. w_n^v takes a value of 1(0) when w_n is $v \in \mathscr{V}$ (is not $v \in \mathscr{V}$). $\boldsymbol{y} = (y_1, y_2, \cdots, y_K)$ is a topic vector into which a document $\boldsymbol{w}$ is categorized, where y_i takes a value of 1(0) when the ith topic is (not) assigned with a document $\boldsymbol{w}$. $I_y \subset \mathscr{Y}$ is a set of topic index i, where y_i takes a value of 1 in $\boldsymbol{y}$. $\sum_{i \in I_y}$ and $\Pi_{i \in I_y}$ denote the sum and product for all i in I_y, respectively. $\Gamma(x)$ is the Gamma function and Ψ is the Psi function(Minka, 2002). A probabilistic generative model for documents with multiple topics models a probability of generating a document $\boldsymbol{w}$ in multiple topics $\boldsymbol{y}$ using model parameter Θ, i.e., models $P(\boldsymbol{w}|\boldsymbol{y}, \Theta)$. A multiple categorization problem is to estimate multiple topics $\boldsymbol{y}^*$ of a document $\boldsymbol{w}^*$ whose topics are unknown. The model parameters are learned by documents $D = \{(\boldsymbol{w_d}, \boldsymbol{y_d})\}_{d=1}^M$, where M is the number of documents.

3 Parametric Mixture Model

In this section, we briefly explain Parametric Mixture Model(PMM)(Ueda, N. and Saito, K., 2002a; Ueda, N. and Saito, K., 2002b).

3.1 Overview

PMM models multiple-topic documents by mixing model parameters of each single topic with an equal mixture ratio, where the model parameter θ_{iv} is the probability that word v is generated from topic i. This is because it is impractical to use model parameter corresponding to multiple topics whose number is $2^K - 1$(all combination of K topics). PMM achieved more useful results than machine learning methods such as Naive Bayes, SVM, K-NN and Neural Networks (Ueda, N. and Saito, K., 2002a; Ueda, N. and Saito, K., 2002b).

3.2 Formulation

PMM employs a BOW representation and is formulated as follows.

$$P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\theta}) = \Pi_{v=1}^{V} (\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}))^{x_v} \tag{1}$$

$\boldsymbol{\theta}$ is a $K \times V$ matrix whose element is $\theta_{iv} = P(v|y_i = 1)$. $\varphi(v, \boldsymbol{y}, \boldsymbol{\theta})$ is the probability that word v is generated from multiple topics $\boldsymbol{y}$ and is defined as the linear sum of $h_i(\boldsymbol{y})$ and θ_{iv} as follows: $\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}) = \sum_{i=1}^K h_i(\boldsymbol{y})\theta_{iv}$

$h_i(\boldsymbol{y})$ is a mixture ratio corresponding to topic i and is formulated as follows:

$$h_i(\boldsymbol{y}) = \frac{y_i}{\sum_{j=1}^K y_j}, \sum_{i=1}^K h_i(\boldsymbol{y}) = 1.$$
$$\text{(if } y_i = 0\text{, then } h_i(\boldsymbol{y}) = 0)$$

3.3 Learning Algorithm of Model Parameter

The learning algorithm of model parameter $\boldsymbol{\theta}$ in PMM is an iteration method similar to the EM algorithm. Model parameter $\boldsymbol{\theta}$ is estimated by maximizing $\Pi_{d=1}^M P(\boldsymbol{w_d}|\boldsymbol{y_d}, \boldsymbol{\theta})$ in training documents $D = \{(\boldsymbol{w_d}, \boldsymbol{y_d})\}_{d=1}^M$. Function g corresponding to a document d is introduced as follows:

$$g_{iv}^d(\boldsymbol{\theta}) = \frac{h(\boldsymbol{y_d})\theta_{iv}}{\sum_{j=1}^K h_j(\boldsymbol{y_d})\theta_{jv}} \tag{2}$$

The parameters are updated along with the following formula.

$$\theta_{iv}^{(t+1)} = \frac{1}{C}(\sum_d^M x_{dv} g_{iv}^d(\boldsymbol{\theta}^{(t)}) + \zeta - 1) \tag{3}$$

x_{dv} is the frequency of word v in document d. C is the normalization term for $\sum_{v=1}^V \theta_{iv} = 1$. ζ is a smoothing parameter that is *Laplace smoothing* when ζ is set to two. In this paper, ζ is set to two as the original paper.

4 Proposed Model

In this section, firstly, we mention the problem related to PMM. Then, we explain our solution of the problem by proposing a new model.

4.1 Overview

PMM estimates model parameter $\boldsymbol{\theta}$ assuming that all of mixture ratios of single topic are equal. It is our intuition that each document can sometimes be more weighted to some topics than to the rest of the assigned topics. If the topic weightings are averaged over all biases in the whole document set, they could be canceled. Therefore, model parameter $\boldsymbol{\theta}$ learned by PMM can be reasonable over the whole of documents.

However, if we compute the probability of generating an individual document, a document-specific topic weight bias on mixture ratio is to be considered. The proposed model takes into account this document-specific bias by assuming that mixture ratio vector $\boldsymbol{\pi}$ follows Dirichlet distribution. This is because we assume the sum of the element in vector $\boldsymbol{\pi}$ is one and each element π_i is nonnegative. Namely, the proposed model assumes model parameter of multiple topics as a mixture of model parameter on each single topic with mixture ratio following Dirichlet distribution. Concretely, given a document $\boldsymbol{w}$ and multiple topics $\boldsymbol{y}$, it estimates a posterior probability distribution $P(\boldsymbol{\pi}|\boldsymbol{x}, \boldsymbol{y})$ by Bayesian inference. For convenience, the proposed model is called PDMM(Parametric Dirichlet Mixture Model).

In Figure 1, the mixture ratio(bias) $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3), \sum_{i=1}^{3} \pi_i = 1, \pi_i > 0$ of three topics is expressed in 3-dimensional real space $\boldsymbol{R}^3$. The mixture ratio(bias) $\boldsymbol{\pi}$ constructs 2D-simplex in $\boldsymbol{R}^3$. One point on the simplex indicates one mixture ratio $\boldsymbol{\pi}$ of the three topics. That is, the point indicates multiple topics with the mixture ratio. PMM generates documents assuming that each mixture ratio is equal. That is, PMM generates only documents with multiple topics that indicates the center point of the 2D-simplex in Figure 1. On the contrary, PDMM generates documents assuming that mixture ratio $\boldsymbol{\pi}$ follows Dirichlet distribution. That is, PDMM can generate documents with multiple topics whose weights can be generated by Dirichlet distribution.

4.2 Formulation

PDMM is formulated as follows:

$$P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$$

$$= \int P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y}) \Pi_{v=1}^{V} (\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\pi}))^{x_v} d\boldsymbol{\pi} \qquad (4)$$

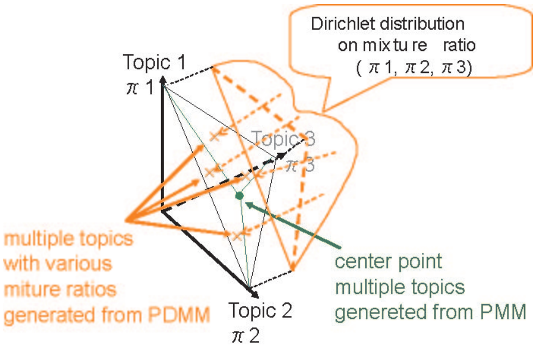


Figure 1: Topic Simplex for Three Topics

$\boldsymbol{\pi}$ is a vector whose element is $\pi_i (i \in I_y)$. π_i is a mixture ratio(bias) of model parameter corresponding to single topic i where $\pi_i > 0, \sum_{i \in I_y} \pi_i = 1$. π_i can be considered as a probability of topic i , i.e., $\pi_i = P(y_i = 1|\boldsymbol{\pi})$. $P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y})$ is a prior distribution of $\boldsymbol{\pi}$ whose index i is an element of I_y, i.e., $i \in I_y$. We use Dirichlet distribution as the prior. $\boldsymbol{\alpha}$ is a parameter vector of Dirichlet distribution corresponding to $\pi_i (i \in I_y)$. Namely, the formulation is as follows.

$$P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y}) = \frac{\Gamma(\sum_{i \in I_y} \alpha_i)}{\Pi_{i \in I_y} \Gamma(\alpha_i)} \Pi_{i \in I_y} \pi_i^{\alpha_i - 1} \qquad (5)$$

$\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\pi})$ is the probability that word v is generated from multiple topics $\boldsymbol{y}$ and is denoted as a linear sum of $\pi_i (i \in I_y)$ and $\theta_{iv} (i \in I_y)$ as follows.

$$\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{i \in I_y} \pi_i \theta_{iv} \qquad (6)$$

$$= \sum_{i \in I_y} P(y_i = 1|\boldsymbol{\pi}) P(v|y_i = 1, \theta) \qquad (7)$$

4.3 Variational Bayes Method for Estimating Mixture Ratio

This section explains a method to estimate the posterior probability distribution $P(\boldsymbol{\pi}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$ of a document-specific mixture ratio. Basically, $P(\boldsymbol{\pi}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$ is obtained by Bayes theorem using Eq.(4). However, that is computationally impractical because a complicated integral computation is needed. Therefore we estimate an approximate distribution of $P(\boldsymbol{\pi}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$ using Variational Bayes Method(H.Attias, 1999). The concrete explanation is as follows

Use Eqs.(4)(7).

$$P(\boldsymbol{w}, \boldsymbol{\pi}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta}) =$$
$$P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y})\Pi_{v=1}^{V}(\sum_{i\in I_y} P(y_i = 1|\boldsymbol{\pi})P(v|y_i = 1, \theta))^{x_v}$$

Transform document expression of above equation into words vector $\boldsymbol{w} = (w_1, w_2, \cdots, w_N)$.

$$P(\boldsymbol{w}, \boldsymbol{\pi}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta}) =$$
$$P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y})\Pi_{n=1}^{N} \sum_{i_n\in I_y} P(y_{i_n} = 1|\boldsymbol{\pi})P(w_n|y_{i_n} = 1, \theta)$$

By changing the order of $\sum$ and Π, we have

$$P(\boldsymbol{w}, \boldsymbol{\pi}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta}) =$$
$$P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y}) \sum_{\boldsymbol{i}\in I_y^N} \Pi_{n=1}^{N} P(y_{i_n} = 1|\boldsymbol{\pi})P(w_n|y_{i_n} = 1, \theta)$$
$$(\sum_{\boldsymbol{i}\in I_y^N} \equiv \sum_{i_1\in I_y}\sum_{i_2\in I_y}\cdots\sum_{i_N\in I_y})$$

Express $y_{i_n} = 1$ as $z_n = i$.

$$P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta}) =$$
$$\int \sum_{\boldsymbol{z}\in I_y^N} P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y})\Pi_{n=1}^{N} P(z_n|\boldsymbol{\pi})P(w_n|z_n, \theta)d\boldsymbol{\pi}$$
$$(\sum_{\boldsymbol{z}\in I_y^N} \equiv \sum_{z_1\in I_y}\sum_{z_2\in I_y}\cdots\sum_{z_N\in I_y}) \qquad (8)$$

Eq.(8) is regarded as Eq.(4) rewritten by introducing a new latent variable $\boldsymbol{z} = (z_1, z_2, \cdots, z_N)$.

$$P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta}) = \int \sum_{\boldsymbol{z}\in I_y^N} P(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})d\boldsymbol{\pi} \quad (9)$$

Use Eqs.(8)(9)

$$P(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$$
$$= P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y})\Pi_{n=1}^{N} P(z_n|\boldsymbol{\pi})P(w_n|z_n, \theta) \qquad (10)$$

Hereafter, we explain Variational Bayes Method for estimating an approximate distribution of $P(\boldsymbol{\pi}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$ using Eq.(10). This approach is the same as LDA(D.M.Blei et al., 2001; D.M.Blei et al., 2003). The approximate distribution is assumed to be $Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})$. The following assumptions are introduced.

$$Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi}) = Q(\boldsymbol{\pi}|\gamma)Q(\boldsymbol{z}|\boldsymbol{\phi}) \qquad (11)$$
$$Q(\boldsymbol{\pi}|\gamma) = \frac{\Gamma(\sum_{i\in I_y}\gamma_i)}{\Pi_{i\in I_y}\Gamma(\gamma_i)}\Pi_{i\in I_y}\pi_i^{\gamma_i - 1} \qquad (12)$$
$$Q(\boldsymbol{z}|\boldsymbol{\phi}) = \Pi_{n=1}^{N} Q(z_n|\boldsymbol{\phi}) \qquad (13)$$
$$Q(z_n|\boldsymbol{\phi}) = \Pi_{i=1}^{K}(\phi_{ni})^{z_n^i} \qquad (14)$$

$Q(\boldsymbol{\pi}|\gamma)$ is Dirichlet distribution where γ is its parameter. $Q(z_n|\boldsymbol{\phi})$ is Multinomial distribution where ϕ_{ni} is its parameter and indicates the probability that the nth word of a document is topic i, i.e. $P(y_{i_n} = 1)$. z_n^i is a value of 1(0) when z_n is (not) i. According to Eq.(11), $Q(\boldsymbol{\pi}|\gamma)$ is regarded as an approximate distribution of $P(\boldsymbol{\pi}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$

The log likelihood of $P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$ is derived as follows.

$$\log P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$$
$$= \int \sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})d\boldsymbol{\pi} \log P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$$
$$= \int \sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi}) \log \frac{P(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})}{Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})}d\boldsymbol{\pi}$$
$$+ \int \sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi}) \log \frac{Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})}{P(\boldsymbol{\pi}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})}d\boldsymbol{\pi}$$

$$\log P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta}) = \mathscr{F}[Q] + KL(Q, P) \qquad (15)$$

$$\mathscr{F}[Q] = \int \textstyle\sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi}) \log \frac{P(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})}{Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})}d\boldsymbol{\pi}$$
$$KL(Q, P) = \int \textstyle\sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi}) \log \frac{Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})}{P(\boldsymbol{\pi}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})}d\boldsymbol{\pi}$$

$KL(Q, P)$ is the Kullback-Leibler Divergence that is often employed as a distance between probability distributions. Namely, $KL(Q, P)$ indicates a distance between $Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})$ and $P(\boldsymbol{\pi}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$. $\log P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$ is not relevant to $Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})$. Therefore, $Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})$ that maximizes $\mathscr{F}[Q]$ minimizes $KL(Q, P)$, and gives a good approximate distribution of $P(\boldsymbol{\pi}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\theta})$.

We estimate $Q(\boldsymbol{\pi}, \boldsymbol{z}|\gamma, \boldsymbol{\phi})$, concretely its parameter $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$, by maximizing $\mathscr{F}[Q]$ as follows.

Using Eqs.(10)(11).

$$\mathscr{F}[Q] = \int Q(\boldsymbol{\pi}|\gamma) \log P(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{y})d\boldsymbol{\theta} \qquad (16)$$
$$+ \int \sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{\pi}|\gamma)Q(\boldsymbol{z}|\boldsymbol{\phi}) \log \Pi_{n=1}^{N} P(z_n|\boldsymbol{\pi})d\boldsymbol{\theta} \quad (17)$$
$$+ \sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{z}|\boldsymbol{\phi}) \log \Pi_{n=1}^{N} P(w_n|z_n, \theta) \qquad (18)$$
$$- \int Q(\boldsymbol{\pi}|\gamma) \log Q(\boldsymbol{\pi}|\gamma)d\boldsymbol{\theta} \qquad (19)$$
$$- \sum_{\boldsymbol{z}\in I_y^N} Q(\boldsymbol{z}|\boldsymbol{\phi}) \log Q(\boldsymbol{z}|\boldsymbol{\phi}) \qquad (20)$$

$$\begin{aligned} = \quad & \log\Gamma(\textstyle\sum_{i\in I_y}\alpha_j) - \sum_{i\in I_y}\log\Gamma(\alpha_i) \\ & + \textstyle\sum_{i\in I_y}(\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{j\in I_y}\gamma_j)) \end{aligned} \tag{21}$$

$$+ \quad \sum_{n=1}^{N}\sum_{i\in I_y}\phi_{ni}(\Psi(\gamma_i) - \Psi(\sum_{j\in I_y}\gamma_j)) \tag{22}$$

$$+ \quad \sum_{n=1}^{N}\sum_{i\in I_y}\sum_{j=1}^{V}\phi_{ni}w_n^j \log\theta ij \tag{23}$$

$$\begin{aligned} - \quad & \log\Gamma(\sum_{j\in I_y}\gamma_j) + \sum_{i\in I_y}\log\Gamma(\sum_{j\in I_y}\gamma_j) \\ & -\sum_{i\in I_y}(\gamma_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{j\in I_y}\gamma_j)) \end{aligned} \tag{24}$$

$$- \quad \sum_{n=1}^{N}\sum_{i\in I_y}\phi_{ni}\log\phi_{ni} \tag{25}$$

$\mathscr{F}[Q]$ is known to be a function of γ_i and ϕ_{ni} from Eqs.(21) through (25). Then we only need to resolve the maximization problem of nonlinear function $\mathscr{F}[Q]$ with respect to γ_i and ϕ_{ni}. In this case, the maximization problem can be resolved by Lagrange multiplier method.

First, regard $\mathscr{F}[Q]$ as a function of γ_i, which is denoted as $\mathscr{F}[\gamma_i]$. Then ,γ_i does not have constraints. Therefore we only need to find the following γ_i, where $\frac{\partial\mathscr{F}[\gamma_i]}{\partial\gamma_i} = 0$. The resultant γ_i is expressed as follows.

$$\gamma_i = \alpha_i + \sum_{n=1}^{N}\phi_{ni} \quad (i \in I_y) \tag{26}$$

Second, regard $\mathscr{F}[Q]$ as a function of ϕ_{ni}, which is denoted as $\mathscr{F}[\phi_{ni}]$. Then, considering the constraint that $\sum_{i\in I_y}\phi_{ni} = 1$, Lagrange function $L[\phi_{ni}]$ is expressed as follows:

$$L[\phi_{ni}] = \mathscr{F}[\phi_{ni}] + \lambda(\sum_{i\in I_y}\phi_{ni} - 1) \tag{27}$$

λ is a so-called Lagrange multiplier.

We find the following ϕ_{ni} where $\frac{\partial L[\phi_{ni}]}{\partial\phi ni} = 0$.

$$\phi_{ni} = \frac{\theta_{iw_n}}{C}\exp(\Psi(\gamma_i) - \Psi(\sum_{j\in I_y}\gamma_j)) \ \ (i \in I_y)) \tag{28}$$

C is a normalization term. By Eqs.(26)(28), we obtain the following updating formulas of γ_i and ϕ_{ni}.

$$\gamma_i^{(t+1)} = \alpha_i + \sum_{n=1}^{N}\phi_{ni}^{(t)} \quad (i \in I_y) \tag{29}$$

$$\phi_{ni}^{(t+1)} = \frac{\theta_{iw_n}}{C}\exp(\Psi(\gamma_i^{(t+1)}) - \Psi(\sum_{j\in I_y}\gamma_j^{(t+1)})) \tag{30}$$

Using the above updating formulas , we can estimate parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$, which are specific to a document $\boldsymbol{w}$ and topics $\boldsymbol{y}$. Last of all , we show a pseudo code :$vb(\boldsymbol{w},\boldsymbol{y})$ which estimates γ and ϕ. In addition , we regard $\boldsymbol{\alpha}$, which is a parameter of a prior distribution of $\boldsymbol{\pi}$, as a vector whose elements are all one. That is because Dirichlet distribution where each parameter is one becomes Uniform distribution.

• Variational Bayes Method for PDMM————function vb($\boldsymbol{w},\boldsymbol{y}$):

```
1. Initialize $\alpha_i \leftarrow 1 \ \forall i \in I_y$
2. Compute $\boldsymbol{\gamma^{(t+1)}},\boldsymbol{\phi^{(t+1)}}$ using Eq.(29)(30)
3. if $\parallel \boldsymbol{\gamma^{(t+1)}} - \boldsymbol{\gamma^{(t)}} \parallel < \epsilon$
   & $\parallel \boldsymbol{\phi^{(t+1)}} - \boldsymbol{\phi^{(t)}} \parallel < \epsilon$
4. then return $(\boldsymbol{\gamma^{(t+1)}},\boldsymbol{\phi^{(t+1)}})$ and halt
5. else $t \leftarrow t+1$ and goto step (2)
```

4.4 Computing Probability of Generating Document

PMM computes a probability of generating a document $\boldsymbol{w}$ on topics $\boldsymbol{y}$ and a set of model parameter Θ as follows:

$$P(\boldsymbol{w}|\boldsymbol{y},\Theta) = \Pi_{v=1}^{V}(\varphi(v,\boldsymbol{y},\boldsymbol{\theta}))^{x_v} \tag{31}$$

$\varphi(v,\boldsymbol{y},\boldsymbol{\theta})$ is the probability of generating a word v on topics $\boldsymbol{y}$ that is a mixture of model parameter $\theta_{iv}(i \in I_y)$ with an equal mixture ratio. On the other hand, PDMM computes the probability of generating a word v on topics $\boldsymbol{y}$ using $\theta_{iv}(i \in I_y)$ and an approximate posterior distribution $Q(\pi|\boldsymbol{\gamma})$ as follows:

$$\begin{aligned}\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\gamma}) & \\ &= \int (\sum_{i \in I_y} \pi_i \theta_{iv}) Q(\boldsymbol{\pi}|\boldsymbol{\gamma}) d\boldsymbol{\pi} & (32) \\ &= \sum_{i \in I_y} \int \pi_i Q(\pi|\boldsymbol{\gamma}) d\boldsymbol{\pi} \theta_{iv} & (33) \\ &= \sum_{i \in I_y} \tilde{\pi}_i \theta_{iv} & (34)\end{aligned}$$

$\tilde{\pi}_i = \int \pi_i Q(\pi|\boldsymbol{\gamma}) d\boldsymbol{\pi} = \frac{\gamma_i}{\sum_{j \in I_y} \gamma_j}$ (C.M.Bishop, 2006)

The above equation regards the mixture ratio of topics $\boldsymbol{y}$ of a document $\boldsymbol{w}$ as the expectation $\tilde{\pi}_i (i \in I_y)$ of $Q(\pi|\boldsymbol{\gamma})$. Therefore, a probability of generating $\boldsymbol{w}$ $P(\boldsymbol{w}|\boldsymbol{y}, \Theta)$ is computed with $\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\gamma})$ estimated in the following manner:

$$P(\boldsymbol{w}|\boldsymbol{y}, \Theta) = \Pi_{v=1}^{V} (\varphi(v, \boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\gamma})))^{x_v} \qquad (35)$$

4.5 Algorithm for Estimating Multiple Topics of Document

PDMM estimates multiple topics $\boldsymbol{y}^*$ maximizing a probability of generating a document $\boldsymbol{w}^*$, i.e., Eq.(35). This is the 0-1 integer problem(i.e., NP-hard problem), so PDMM uses the same approximate estimation algorithm as PMM does. But it is different from PMM's estimation algorithm in that it estimates the mixture ratios of topics $\boldsymbol{y}$ by Variational Bayes Method as shown by vb(w,y) at step 6 in the following pseudo code of the estimation algorithm:

```
• Topics Estimation Algorithm
function prediction($\boldsymbol{w}$):
1. Initialize $S \leftarrow \{1, 2, \cdots\}, y_i \leftarrow 0$ for $i(1, 2 \cdots, K)$
2. $v_{max} \leftarrow -\infty$
3. while $S$ is not empty do
4.     foreach $i \in S$ do
5.         $y_i \leftarrow 1, y_{j \in S \setminus i} \leftarrow 0$
6.         Compute $\boldsymbol{\gamma}$ by vb($\boldsymbol{w}, \boldsymbol{y}$)
7.         $v(i) \leftarrow P(\boldsymbol{w}|\boldsymbol{y})$
8.     end foreach
9.     $i^* \leftarrow$ argmax $v(i)$
10.    if $v(i^*) > v_{max}$
11.        $y_{i^*} \leftarrow 1, S \leftarrow S \setminus i^*, v_{max} \leftarrow v(i^*)$
12.    else
13.        return $\boldsymbol{y}$ and halt
```

5 Evaluation

We evaluate the proposed model by using F-measure of multiple topics categorization problem.

5.1 Dataset

We use MEDLINE[1] as a dataset. In this experiment, we use five thousand abstracts written in English. MEDLINE has a metadata set called MeSH Term. For example, each abstract has MeSH Terms such as RNA Messenger and DNA-Binding Proteins. MeSH Terms are regarded as multiple topics of an abstract. In this regard, however, we use MeSH Terms whose frequency are medium(100-999). We did that because the result of experiment can be overly affected by such high frequency terms that appear in almost every abstract and such low frequency terms that appear in very few abstracts. In consequence, the number of topics is 88. The size of vocabulary is 46,075. The proportion of documents with multiple topics on the whole dataset is 69.8%, i.e., that of documents with single topic is 30.2%. The average of the number of topics of a document is 3.4. Using TreeTagger[2], we lemmatize every word. We eliminate stop words such as articles and be-verbs.

5.2 Result of Experiment

We compare F-measure of PDMM with that of PMM and other models.

F-measure(F) is as follows:
$F = \frac{2PR}{P+R}, P = \frac{|N_r \cap N_e|}{|N_e|}, R = \frac{|N_r \cap N_e|}{|N_r|}$.

N_r is a set of relevant topics . N_e is a set of estimated topics. A higher F-measure indicates a better ability to discriminate topics. In our experiment, we compute F-measure in each document and average the F-measures throughout the whole document set.

We consider some models that are distinct in learning model parameter $\boldsymbol{\theta}$. **PDMM** learns model parameter $\boldsymbol{\theta}$ by the same learning algorithm as PMM. **NBM** learns model parameter $\boldsymbol{\theta}$ by Naive Bayes learning algorithm. The parameters are updated according to the following formula: $\theta_{iv} = \frac{M_{iv}+1}{C}$. M_{iv} is the number of training documents where a word v appears in topic i. C is normalization term for $\sum_{v=1}^{V} \theta_{iv} = 1$.

[1] http://www.nlm.nih.gov/pubs/factsheets/medline.html

[2] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

The comparison of these models with respect to F-measure is shown in Figure 2. The horizontal axis is the proportion of test data of dataset(5,000 abstracts). For example, 2% indicates that the number of documents for learning model is 4,900 and the number of documents for the test is 100. The vertical axis is F-measure. In each proportion, F-measure is an average value computed from five pairs of training documents and test documents randomly generated from dataset.

F-measure of PDMM is higher than that of other methods on any proportion, as shown in Figure 2. Therefore, PDMM is more effective than other methods on multiple topics categorization.

Figure 3 shows the comparison of models with respect to F-measure, changing proportion of multiple topic document for the whole dataset. The proportion of document for learning and test are 40% and 60%, respectively. The horizontal axis is the proportion of multiple topic document on the whole dataset. For example, 30% indicates that the proportion of multiple topic document is 30% on the whole dataset and the remaining documents are single topic , that is, this dataset is almost single topic document. In 30%. there is little difference of F-measure among models. As the proportion of multiple topic and single topic document approaches 90%, that is, multiple topic document, the differences of F-measure among models become apparent. This result shows that PDMM is effective in modeling multiple topic document.

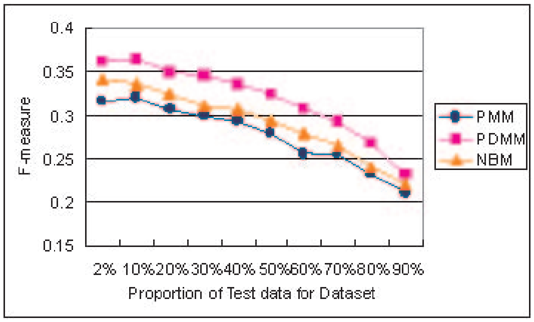


Figure 2: F-measure Results

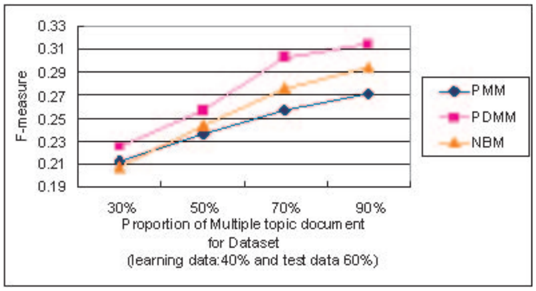


Figure 3: F-measure Results changing Proportion of Multiple Topic Document for Dataset

5.3 Discussion

In the results of experiment described in section 5.2, PDMM is more effective than other models in multiple-topic categorization. If the topic weightings are averaged over all biases in the whole of training documents, they could be canceled. This cancellation can lead to the result that model parameter $\boldsymbol{\theta}$ learned by PMM is reasonable over the whole of documents. Moreover, PDMM computes the probability of generating a document using a mixture of model parameter, estimating the mixture ratio of topics. This estimation of the mixture ratios, we think, is the key factor to achieve the results better than other models. In addition, the estimation of a mixture ratio of topics can be effective from the perspective of extracting features of a document with multiple topics. A mixture ratio of topics assigned to a document is specific to the document. Therefore, the estimation of the mixture ratio of topics is regarded as a projection from a word-frequency space of $\mathscr{Q}^V$ where $\mathscr{Q}$ is a set of integer number to a mixture ratio space of topics $[0, 1]^K$ in a document. Since the size of vocabulary is much more than that of topics, the estimation of the mixture ratio of topics is regarded as a dimension reduction and an extraction of features in a document. This can lead to analysis of similarity among documents with multiple topics. For example, the estimated mixture ratio of topics [Comparative Study]C[Apoptosis] and [Models,Biological] in one MEDLINE abstract is 0.656C0.176 and 0.168, respectively. This ratio can be a feature of this document.

Moreover, we can obtain another interesting results as follows. The estimation of mixture ratios of topics uses parameter γ in section 4.3. We obtain interesting results from another parameter ϕ that needs to estimate γ. ϕ_{ni} is specific to a document. A

Table 1: Word List of Document X whose Topics are [Female], [Male] and [Biological Markers]

Ranking	Top10	Ranking	Bottom10
1(37)	biomarkers	67(69)	indicate
2(19)	Fusarium	68(57)	problem
3(20)	non-Gaussian	69(45)	use
4(21)	Stachybotrys	70(75)	%
5(7)	chrysogenum	71(59)	correlate
6(22)	Cladosporium	72(17)	population
7(3)	mould	73(15)	healthy
8(35)	Aspergillus	7433)	response
9(23)	dampness	75(56)	man
10(24)	1SD	76(64)	woman

Table 2: Word List of Document X whose Topics are [Rats], [Child] and [Incidence]

Ranking	Top10	Ranking	Bottom10
1(69)	indicate	67(56)	man
2(63)	relate	68(47)	blot
3(53)	antigen	69(6)	exposure
4(45)	use	70(54)	distribution
5(3)	mould	71(68)	evaluate
6(4)	versicolor	72(67)	examine
7(35)	Aspergillus	73(59)	correlate
8(7)	chrysogenum	74(58)	positive
9(8)	chartarum	75(1)	IgG
10(9)	herbarum	76(60)	adult

ϕ_{ni} indicates the probability that a word w_n belongs to topic i in a document. Therefore we can compute the entropy on w_n as follows:

$$entropy(w_n) = \sum_{i=1}^{K} \phi_{ni} \log(\phi_{ni})$$

We rank words in a document by this entropy. For example, a list of words in ascending order of the entropy in document X is shown in Table 1. A value in parentheses is a ranking of words in decending order of TF-IDF($= tf \cdot \log(M/df)$,where tf is term frequency in a test document, df is document frequency and M is the number of documents in the set of doucuments for learning model parameters) (Y. Yang and J. Pederson, 1997) . The actually assigned topics are [Female] , [Male] and [Biological Markers], where each estimated mixture ratio is 0.499 , 0.460 and 0.041, respectively.

The top 10 words seem to be more technical than the bottom 10 words in Table 1. When the entropy of a word is lower, the word is more topic-specific oriented, i.e., more technical . In addition, this ranking of words depends on topics assigned to a document. When we assign randomly chosen topics to the same document, generic terms might be ranked higher. For example, when we rondomly assign the topics [Rats], [Child] and [Incidence], generic terms such as ”use” and ”relate” are ranked higher as shown in Table 2. The estimated mixture ratio of [Rats], [Child] and [Incidence] is 0.411, 0.352 and 0.237, respectively.

For another example, a list of words in ascending order of the entropy in document Y is shown in Table 3. The actually assigned topics are Female, Animals, Pregnancy and Glucose.. The estimated mixture ratio of [Female], [Animals] ,[Pregnancy] and [Glucose] is 0.442, 0.437, 0.066 and 0.055, respectively In this case, we consider assigning sub topics of actual topics to the same document Y.

Table 4 shows a list of words in document Y assigned with the sub topics [Female] and [Animals]. The estimated mixture ratio of [Female] and [Animals] is 0.495 and 0.505, respectively. Estimated mixture ratio of topics is chaged. It is interesting that [Female] has higher mixture ratio than [Animals] in actual topics but [Female] has lower mixture ratio than [Animals] in sub topics [Female] and [Animals]. According to these different mixture ratios, the ranking of words in docment Y is changed.

Table 5 shows a list of words in document Y assigned with the sub topics [Pregnancy] and [Glucose]. The estimated mixture ratio of [Pregnancy] and [Glucose] is 0.502 and 0.498, respectively. It is interesting that in actual topics, the ranking of gglucose-insulinh and ”IVGTT” is high in document Y but in the two subset of actual topics, gglucose-insulinh and ”IVGTT” cannot be find in Top 10 words.

The important observation known from these examples is that this ranking method of words in a document can be assosiated with topics assigned to the document. ϕ depends on γ seeing Eq.(28). This is because the ranking of words depends on assigned topics, concretely, mixture ratios of assigned topics. TF-IDF computed from the whole documents cannot have this property. Combined with existing the extraction method of keywords, our model has the potential to extract document-specific keywords using information of assigned topics.

Table 3: Word List of Document Y whose Actual Topics are [Femaile],[Animals],[Pregnancy] and [Glucose]

Ranking	Top 10	Ranking	Bottom 10
1(2)	glucose-insulin	94(93)	assess
2(17)	IVGTT	95(94)	indicate
3(11)	undernutrition	96(74)	CT
4(12)	NR	97(28)	%
5(13)	NRL	98(27)	muscle
6(14)	GLUT4	99(85)	receive
7(56)	pregnant	100(80)	status
8(20)	offspring	101(100)	protein
9(31)	pasture	102(41)	age
10(32)	singleton	103(103)	conclusion

Table 4: Word List of Document Y whose Topics are [Femaile]and [Animals]

Ranking	Top 10	Ranking	Bottom 10
1(31)	pasture	94(65)	insulin
2(32)	singleton	95(76)	reduced
3(33)	insulin-signaling	96(27)	muscle
4(34)	CS	97(74)	CT
5(35)	euthanasia	98(68)	feed
6(36)	humane	99(100)	protein
7(37)	NRE	100(80)	status
8(38)	110-term	101(85)	receive
9(50)	insert	102(41)	age
10(11)	undernutrition	103(103)	conclusion

Table 5: Word List of Document Y whose Topics are [Pregnancy]and [Glucose]

Ranking	Top 10	Ranking	Bottom 10
1(84)	mass	94(18)	IVGTT
2(74)	CT	95(72)	metabolism
3(26)	requirement	96(73)	metabolic
4(45)	intermediary	97(57)	pregnant
5(50)	insert	98(58)	prenatal
6(53)	feeding	99(59)	fetal
7(55)	nutrition	100(3)	gestation
8(61)	nutrient	101(20)	offspring
9(31)	pasture	102(65)	insulin
10(32)	singleton	103(16)	glucose

6 Concluding Remarks

We proposed and evaluated a novel probabilistic generative models, PDMM, to deal with multiple-topic documents. We evaluated PDMM and other models by comparing F-measure using MEDLINE corpus. The results showed that PDMM is more effective than PMM. Moreover, we indicate the potential of the proposed model that extracts document-specific keywords using information of assigned topics.

Acknowledgement This research was funded in part by MEXT Grant-in-Aid for Scientific Research on Priority Areas "i-explosion" in Japan.

References

H.Attias 1999. Learning parameters and structure of latent variable models by variational Bayes. *in Proc of Uncertainty in Artificial Intelligence*.

C.M.Bishop 2006. Pattern Recognition And Machine Learning (Information Science and Statistics), p.687. *Springer-Verlag*.

D.M. Blei, Andrew Y. Ng, and M.I. Jordan. 2001. Latent Dirichlet Allocation. *Neural Information Processing Systems* 14.

D.M. Blei, Andrew Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, vol.3, pp.993-1022.

Minka 2002. Estimating a Dirichlet distribution. *Technical Report*.

Y.W.Teh, M.I.Jordan, M.J.Beal, and D.M.Blei. 2003. Hierarchical dirichlet processes. *Technical Report* 653, Department Of Statistics, UC Berkeley.

Ueda, N. and Saito, K. 2002. Parametric mixture models for multi-topic text. *Neural Information Processing Systems* 15.

Ueda, N. and Saito, K. 2002. Singleshot detection of multi-category text using parametric mixture models. *ACM SIG Knowledge Discovery and Data Mining*.

Y. Yang and J. Pederson 1997. A comparative study on feature selection in text categorization. *Proc. International Conference on Machine Learning*.

Smooth Bilingual N-gram Translation

Holger Schwenk
LIMSI-CNRS, BP 133
91403 Orsay cedex, FRANCE
schwenk@lismi.fr

Marta R. Costa-jussà and ***José A.R. Fonollosa***
UPC - TALP
Barcelona 08034, Spain
{mruiz,adrian}@gps.tsc.upc.edu

Abstract

We address the problem of smoothing translation probabilities in a bilingual N-gram-based statistical machine translation system. It is proposed to project the bilingual tuples onto a continuous space and to estimate the translation probabilities in this representation. A neural network is used to perform the projection and the probability estimation.

Smoothing probabilities is most important for tasks with a limited amount of training material. We consider here the BTEC task of the 2006 IWSLT evaluation. Improvements in all official automatic measures are reported when translating from Italian to English. Using a continuous space model for the translation model and the target language model, an improvement of 1.5 BLEU on the test data is observed.

1 Introduction

The goal of statistical machine translation (SMT) is to produce a target sentence $\mathbf{e}$ from a source sentence $\mathbf{f}$. Among all possible target language sentences the one with the highest probability is chosen:

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \Pr(\mathbf{f}|\mathbf{e}) \Pr(\mathbf{e})$$

where $\Pr(\mathbf{f}|\mathbf{e})$ is the translation model and $\Pr(\mathbf{e})$ is the target language model. This approach is usually referred to as the *noisy source-channel* approach in statistical machine translation (Brown et al., 1993).

During the last few years, the use of context in SMT systems has provided great improvements in translation. SMT has evolved from the original word-based approach to phrase-based translation systems (Och et al., 1999; Koehn et al., 2003). A phrase is defined as a group of source words $\tilde{\mathbf{f}}$ that should be translated together into a group of target words $\tilde{\mathbf{e}}$. The translation model in phrase-based systems includes the phrase translation probabilities in both directions, i.e. $P(\tilde{\mathbf{e}}|\tilde{\mathbf{f}})$ and $P(\tilde{\mathbf{f}}|\tilde{\mathbf{e}})$.

The use of a maximum entropy approach simplifies the introduction of several additional models explaining the translation process :

$$\begin{aligned} \mathbf{e}^* &= \arg\max p(\mathbf{e}|\mathbf{f}) \\ &= \arg\max_{e} \{exp(\sum_i \lambda_i h_i(\mathbf{e}, \mathbf{f}))\} \end{aligned} \quad (1)$$

The feature functions h_i are the system models and the λ_i weights are typically optimized to maximize a scoring function on a development set (Och and Ney, 2002).

The phrase translation probabilities $P(\tilde{\mathbf{e}}|\tilde{\mathbf{f}})$ and $P(\tilde{\mathbf{f}}|\tilde{\mathbf{e}})$ are usually obtained using relative frequency estimates. Statistical learning theory, however, tells us that relative frequency estimates have several drawbacks, in particular high variance and low bias. Phrase tables may contain several millions of entries, most of which appear only once or twice, which means that we are confronted with a data sparseness problem. Surprisingly, there seems to be little work addressing the issue of smoothing of the phrase table probabilities.

On the other hand, smoothing of relative frequency estimates was extensively investigated in the

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 430–438, Prague, June 2007.

area of language modeling. A systematic comparison can be for instance found in (Chen and Goodman, 1999). Language models and phrase tables have in common that the probabilities of rare events may be overestimated. However, in language modeling probability mass must be redistributed in order to account for the unseen n-grams. Generalization to unseen events is less important in phrase-based SMT systems since the system searches only for the best segmentation and the best matching phrase pair among the existing ones.

We are only aware of one work that performs a systematic comparison of smoothing techniques in phrase-based machine translation systems (Foster et al., 2006). Two types of phrase-table smoothing were compared: black-box and glass-box methods. Black-methods do not look inside phrases but instead treat them as atomic objects. By these means, all the methods developed for language modeling can be used. Glass-box methods decompose $P(\tilde{\mathbf{e}}|\tilde{\mathbf{f}})$ into a set of lexical distributions $P(e|\tilde{\mathbf{f}})$. For instance, it was suggested to use IBM-1 probabilities (Och et al., 2004), or other lexical translation probabilities (Koehn et al., 2003; Zens and Ney, 2004). Some form of glass-box smoothing is now used in all state-of-the-art statistical machine translation systems.

Another approach related to phrase table smoothing is the so-called N-gram translation model (Mariño et al., 2006). In this model, bilingual tuples are used instead of the phrase pairs and n-gram probabilities are considered rather than relative frequencies. Therefore, smoothing is obtained using the standard techniques developed for language modeling. In addition, a context dependence of the phrases is introduced. On the other hand, some restrictions on the segmentation of the source sentence must be used. N-gram-based translation models were extensively compared to phrase-based systems on several tasks and typically achieve comparable performance.

In this paper we propose to investigate improved smoothing techniques in the framework of the N-gram translation model. Despite the undeniable success of n-graam back-off models, these techniques have several drawbacks from a theoretical point of view: the words are represented in a discrete space, the vocabulary. This prevents "true interpolation" of the probabilities of unseen n-grams since a change in this word space can result in an arbitrary change of the n-gram probability. An alternative approach is based on a *continuous representation* of the words (Bengio et al., 2003). The basic idea is to convert the word indices to a continuous representation and to use a probability estimator operating in this space. Since the resulting distributions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. Probability estimation and interpolation in a continuous space is mathematically well understood and numerous powerful algorithms are available that can perform meaningful interpolations even when only a limited amount of training material is available. This approach was successfully applied to language modeling in large vocabulary continuous speech recognition (Schwenk, 2007) and to language modeling in phrase-based SMT systems (Schwenk et al., 2006).

In this paper, we investigate whether this approach is useful to smooth the probabilities involved in the bilingual tuple translation model. Reliable estimation of unseen n-grams is very important in this translation model. Most of the trigram tuples encountered in the development or test data were never seen in the training data. N-gram hit rates are reported in the results section of this paper. We report experimental results for the BTEC corpus as used in the 2006 evaluations of the international workshop on spoken language translation IWSLT (Paul, 2006). This task provides a very limited amount of resources in comparison to other tasks like the translation of journal texts (NIST evaluations) or of parliament speeches (TC-STAR evaluations). Therefore, new techniques must be deployed to take the best advantage of the limited resources. Among the language pairs tested in this years evaluation, Italian to English gave the best BLEU results in this year evaluation. The better the translation quality is, the more it is challenging to outperform it without adding more data. We show that a new smoothing technique for the translation model achieves a significant improvement in the BLEU score for a state-of-the-art statistical translation system.

This paper is organized as follows. In the next section we first describe the baseline statistical machine translation systems. Section 3 presents the architecture and training algorithms of the continuous space translation model and section 4 summarizes the experimental evaluation. The paper concludes with a discussion of future research directions.

2 N-gram-based Translation Model

The N-gram-based translation model has been derived from the finite-state perspective; more specifically, from the work of Casacuberta (2001). However, different from it, where the translation model is implemented by using a finite-state transducer, the N-gram-based system implements a bilingual N-gram model. It actually constitutes a language model of bilingual units, referred to as tuples, which approximates the joint probability between source and target languages by using N-grams, such as described by the following equation:

$$p(\mathbf{e},\mathbf{f}) \approx \prod_{k=1}^{K} p((e,f)_k|(e,f)_{k-1},\ldots,(e,f)_{k-4}) \tag{2}$$

where e refers to target, f to source and $(e,f)_k$ to the k^{th} tuple of a given bilingual sentence pair.

Bilingual units (tuples) are extracted from any word-to-word alignment according to the following constraints:

- a monotonic segmentation of each bilingual sentence pairs is produced,
- no word inside the tuple is aligned to words outside the tuple, and
- no smaller tuples can be extracted without violating the previous constraints.

As a consequence of these constraints, only one segmentation is possible for a given sentence pair.

Two important issues regarding this translation model must be considered. First, it often occurs that a large number of single-word translation probabilities are left out of the model. This happens for all words that are always embedded in tuples containing two or more words, then no translation probability for an independent occurrence of these embedded words will exist. To overcome this problem, the tuple trigram model is enhanced by incorporating 1-gram translation probabilities for all the embedded words detected during the tuple extraction step. These 1-gram translation probabilities are computed from the intersection of both the source-to-target and the target-to-source alignments.

The second issue has to do with the fact that some words linked to NULL end up producing tuples with NULL source sides. Since no NULL is actually expected to occur in translation inputs, this type of tuple is not allowed. Any target word that is linked to NULL is attached either to the word that precedes or the word that follows it. To determine this, an approach based on the IBM1 probabilities was used, as described in (Mariño et al., 2006).

2.1 Additional features

The following feature functions were used in the N-gram-based translation system:

- A **target language model**. In the baseline system, this feature consists of a 4-gram back-off model of words, which is trained from the target side of the bilingual corpus.
- A **source-to-target lexicon model and a target-to-source lexicon model**. These feature, which are based on the lexical parameters of the IBM Model 1, provide a complementary probability for each tuple in the translation table.
- A **word bonus function**. This feature introduces a bonus based on the number of target words contained in the partial-translation hypothesis. It is used to compensate for the system's preference for short output sentences.

All these models are combined in the decoder. Additionally, the decoder allows for a non-monotonic search with the following distorsion model.

- A word distance-based **distorsion model**.

$$P(t_1^K) = exp(-\sum_{k=1}^{K} d_k)$$

where d_k is the distance between the first word of the k^{th} tuple (unit), and the last word+1 of the $(k-1)^{th}$ tuple.

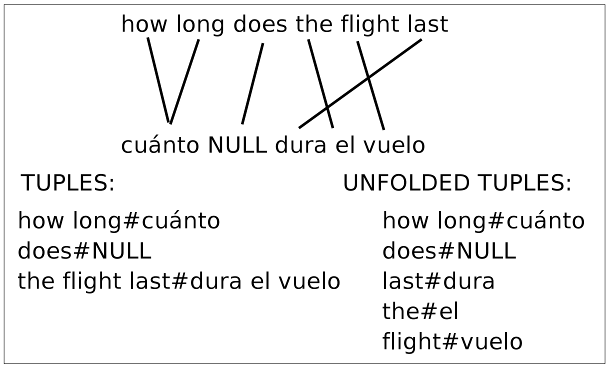


Figure 1: *Comparing regular and unfolded tuples.*

Distance are measured in words referring to the units source side.

To reduce the computational cost we place limits on the search using two parameters: the distortion limit (the maximum distance measured in words that a tuple is allowed to be reordered, m) and the reordering limit (the maximum number of reordering jumps in a sentence, j). Tuples need to be extracted by an unfolding technique (Mariño et al., 2006). This means that the tuples are broken into smaller tuples, and these are sequenced in the order of the target words. In order not to lose the information on the correct order, the decoder performs a non-monotonic search. Figure 1 shows an example of tuple unfolding compared to the monotonic extraction. The unfolding technique produces a different bilingual n-gram language model with reordered source words.

In order to combine the models in the decoder suitably, an optimization tool based on the Simplex algorithm is used to compute log-linear weights for each model.

3 Continuous Space N-gram Models

The architecture of the neural network n-gram model is shown in Figure 2. A standard fully-connected multi-layer perceptron is used. The inputs to the neural network are the indices of the $n-1$ previous units (words or tuples) in the vocabulary $h_j = w_{j-n+1}, \ldots, w_{j-2}, w_{j-1}$ and the outputs are the posterior probabilities of *all* units of the vocabulary:

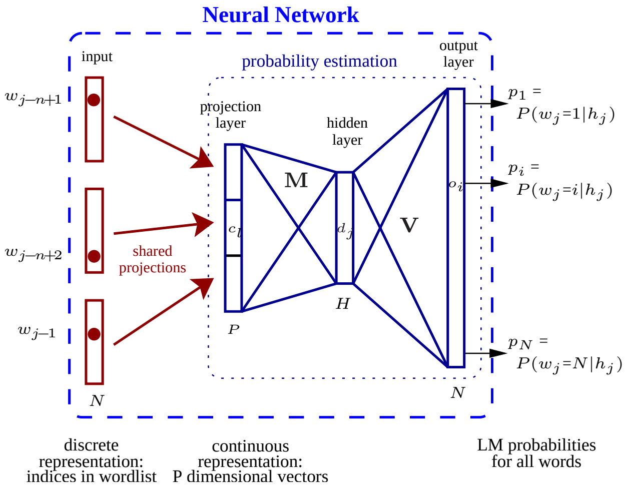


Figure 2: Architecture of the continuous space LM. h_j denotes the context $w_{j-n+1}, \ldots, w_{j-1}$. P is the size of one projection and H,N is the size of the hidden and output layer respectively. When short-lists are used the size of the output layer is much smaller than the size of the vocabulary.

$$P(w_j = i|h_j) \qquad \forall i \in [1, N] \tag{3}$$

where N is the size of the vocabulary. The input uses the so-called 1-of-n coding, i.e., the ith unit of the vocabulary is coded by setting the ith element of the vector to 1 and all the other elements to 0. The ith line of the $N \times P$ dimensional projection matrix corresponds to the continuous representation of the ith unit. Let us denote c_l these projections, d_j the hidden layer activities, o_i the outputs, p_i their softmax normalization, and m_{jl}, b_j, v_{ij} and k_i the hidden and output layer weights and the corresponding biases. Using these notations, the neural network performs the following operations:

$$d_j = \tanh\left(\sum_l m_{jl}\, c_l + b_j\right) \tag{4}$$

$$o_i = \sum_j v_{ij}\, d_j + k_i \tag{5}$$

$$p_i = e^{o_i} \;/\; \sum_{r=1}^{N} e^{o_r} \tag{6}$$

The value of the output neuron p_i corresponds directly to the probability $P(w_j{=}i|h_j)$.

Training is performed with the standard back-propagation algorithm minimizing the following error function:

$$E = \sum_{i=1}^{N} t_i \, log \, p_i + \beta \left(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2 \right) \qquad (7)$$

where t_i denotes the desired output, i.e., the probability should be 1.0 for the next unit in the training sentence and 0.0 for all the other ones. The first part of this equation is the cross-entropy between the output and the target probability distributions, and the second part is a regularization term that aims to prevent the neural network from over-fitting the training data (weight decay). The parameter β has to be determined experimentally. Training is done using a re-sampling algorithm as described in (Schwenk, 2007).

It can be shown that the outputs of a neural network trained in this manner converge to the posterior probabilities. Therefore, the neural network directly minimizes the perplexity on the training data. Note also that the gradient is back-propagated through the projection-layer, which means that the neural network learns the projection of the units onto the continuous space that is best for the probability estimation task.

In general, the complexity to calculate one probability with this basic version of the neural network n-gram model is dominated by the dimension of the output layer since the size of the vocabulary (10k to 64k) is usually much larger than the dimension of the hidden layer (200 to 500). Therefore, in previous applications of the continuous space n-gram model, the output was limited to the s most frequent units, s ranging between 2k and 12k (Schwenk, 2007). This is called a short-list.

	Sents	Words
Train (bitexts)	20k	155.4/166.3k
Dev	489	5.2k
Eval	500	6k

Table 1: Available data in the *supplied resources* of the 2006 IWSLT evaluation.

4 Experimental Evaluation

In this work we report results on the *Basic Traveling Expression Corpus* (BTEC) as used in the 2006 evaluations of the international workshop on spoken language translation (IWSLT). This corpus consists of typical sentences from phrase books for tourists in several languages (Takezawa et al., 2002). We report results on the supplied development corpus of 489 sentences and the official test set of the IWSLT’06 evaluation. The main measure is the BLEU score, using seven reference translations. The scoring is case insensitive and punctuations are ignored. Details on the available data are summarized in Table 1. We concentrated first on the translation from Italian to English. All participants in the IWSLT evaluation achieved much better performances for this language pair than for the other considered translation directions. This makes it more difficult to achieve additional improvements.

A non-monotonic search was performed following a local reordering named in Section 2, setting $m = 5$ and $j = 3$. Also we used histogram pruning in the decoder, i.e. the maximum number of hypotheses in a stack is limited to 50.

4.1 Language-dependent preprocessing

Italian contracted prepositions have been separated into preposition + article, such as ’alla’→’a la’, ’degli’→’di gli’ or ’dallo’→’da lo’, among others.

4.2 Model training

The training and development data for the bilingual back-off and neural network translation model were created as follows. Given the alignment of the training parallel corpus, we perform a unique segmentation of each parallel sentence following the criterion of unfolded segmentation seen in Section 2. This segmentation is used in a sequence as training text for building the language model. As an example, given the alignment and the unfold extraction of Figure 1, we obtain the following training sentence:

<s> how_long#cuánto does#NULL last#dura the#el flight#vuelo </s>

The reference bilingual trigram back-off translation model was trained on these bilingual tuples using the SRI LM toolkit (Stolcke, 2002). Different smoothing techniques were tried, and best results were obtained using Good-Turing discounting.

The neural network approach was trained on exactly the same data. A context of two tuples was used (trigram model). The training corpus contains about 21,500 different bilingual tuples. We decided to limit the output of the neural network to the 8k most frequent tuples (short-list). This covers about 90% of the requested tuple n-grams in the training data.

Similar to previous applications, the neural network is not used alone but interpolation is performed to combine several n-gram models. First of all, the neural network and the reference back-off model are interpolated together - this always improved performance since both seem to be complementary. Second, four neural networks with different sizes of the continuous representation were trained and interpolated together. This usually achieves better generalization behavior than training one larger neural network. The interpolation coefficients were calculated by optimizing perplexity on the development data, using an EM procedure. The obtained values are 0.33 for the back-off translation model and about 0.16 for each neural network model respectively. This interpolation is used in all our experiments. For the sake of simplicity we will still call this the continuous space translation model.

Each network was trained independently using early stopping on the development data. Convergence was achieved after about 10 iterations through the training data (less than 20 minutes of processing on a standard Linux machine). The other parameters are as follows:

- Context of two tuples (trigram)
- The dimension of the continuous representation of the tuples were $c =$120,140,150 and 200,
- The dimension of the hidden layer was set to $P = 200$,
- The initial learning rate was 0.005 with an exponential decay,
- The weight decay coefficient was set to $\beta = 0.00005$.

N-gram models are usually evaluated using perplexity on some development data. In our case, i.e. using bilingual tuples as basic units ("words"), it is less obvious if perplexity is a useful measure. Nevertheless, we provide these numbers for completeness. The perplexity on the development data of the trigram back-off translation model is 227.0. This could be reduced to 170.4 using the neural network. It is also very informative to analyze the n-gram hit-rates of the back-off model on the development data: 10% of the probability requests are actually a true trigram, 40% a bigram and about 49% are finally estimated using unigram probabilities. This means that only a limited amount of phrase context is used in the standard N-gram-based translation model. This makes this an ideal candidate to apply the continuous space model since probabilities are interpolated for all possible contexts and never backed-up to shorter contexts.

4.3 Results and analysis

The incorporation of the neural translation model is done using n-best list. Each hypothesis is composed of a sequence of bilingual tuples and the corresponding scores of all the feature functions. Figure 3 shows an example of such an n-best list. The neural trigram translation model is used to replace the scores of the trigram back-off translation model. This is followed by a re-optimization of the coefficients of all feature functions, i.e. maximization of the BLEU score on the development data using the numerical optimization tool CONDOR (Berghen and Bersini, 2005). An alternative would be to add a feature function and to combine both translation models under the log-linear model framework, using maximum BLEU training.

Another open question is whether it might by better to already use the continuous space translation model during decoding. The continuous space model has a much higher complexity than a back-off n-gram. However, this can be heavily optimized when rescoring n-best lists, i.e. by grouping together all calls in the whole n-best list with the same context, resulting in only one forward pass through the neural network. This is more difficult to perform when the continuous space translation model is used during decoding. Therefore, this was not investigated in this work.

spiacente#sorry tutto_occupato#it_’s_full
spiacente#i_’m_sorry tutto_occupato#it_’s_full
spiacente#i_’m_afraid tutto_occupato#it_’s_full
spiacente#sorry tutto#all occupato#busy
spiacente#sorry tutto#all occupato#taken

Figure 3: Example of sentences in the n-best list of bilingual tuples. The special character ’#’ is used to separate the source and target sentence words. Several words in one tuple a grouped together using ’_.’

In all our experiments 1000-best lists were used. In order to evaluate the quality of these n-best lists, an oracle trigram back-off translation model was build on the development data. Rescoring the n-best lists with this translation model resulted in an increase of the BLEU score of about 10 points (see Table 2). While there is an decrease of about 6% for the position dependent word error rate (mWER), a smaller change in the position independent word error rate was observed (mPER). This suggests that most of the alternative translation hypothesis result in word reorderings and not in many alternative word choices. This is one of the major drawbacks of phrase- and N-gram-based translation systems: only translations observed in the training data can be used. There is no generalization to new phrase pairs.

	Back-off	Oracle	**Neural**
BLEU	42.34	52.45	**43.87**
mWER	41.6%	35.6%	**40.3%**
mPER	31.5%	28.2%	**30.7%**

Table 2: Comparison of different N-gram-translation models on the development data.

When the 1000-best lists are rescored with the neural network translation model the BLEU score increases by 1.5 points (42.34 to 43.87). Similar improvements were observed in the word error rates (see Table 2). For comparison, a 4-gram back-off translation model was also built, but no change of the BLEU score was observed. This suggests that careful smoothing is more important than increasing the context when estimating the translation probabilities in an N-gram-based statistical machine translation system.

In previous work, we have investigated the use of the neural network approach to modeling the target language for the IWSLT task (Schwenk et al., 2006). We also applied this technique to this improved N-gram-based translation system. In our implementation, the neural network target 4-gram language model gives an improvement of 1.3 points BLEU on the development data (42.34 to 43.66), in comparison to 1.5 points for the neural translation model (see Table 3).

	Back-off TM+LM	neural TM	neural LM	**neural TM+LM**
BLEU	42.34	43.87	43.66	**44.83**

Table 3: Combination of a neural translation model (TM) and a neural language model (LM). BLEU scores on the development data.

The neural translation and target language model were also applied to the test data, using of course the same feature function coefficients as for the development data. The results are given in Table 4 for all the official measures of the IWSLT evaluation. The new smoothing method of the translation probabilities achieves improvement in all measures. It gives also an additional gain (again in all measures) when used together with a neural target language model. Surprisingly, neural TM and neural LM improvements almost add up: when both techniques are used together, the BLEU scores increases by 1.5 points (36.97 → 38.50). Remember that the reference N-gram-based translation system already uses a local reordering approach.

	Back-off TM+LM	neural TM	neural LM	**neural TM+LM**
BLEU	36.97	37.21	38.04	**38.50**
mWER	48.10	**47.42**	47.83	47.61
mPER	38.21	38.07	37.26	**37.12**
NIST	8.3	8.3	8.6	**8.7**
Meteor	63.16	63.40	64.70	**65.20**

Table 4: Test set scores for the combination of a neural translation model (TM) and a neural language model (LM).

5 Discussion

Phrase-based approaches are the de-facto standard in statistical machine translation. The phrases are extracted automatically from the word alignments of parallel texts, and the different possible translations of a phrase are weighted using relative frequency. This can be problematic when the data is sparse. However, there seems to be little work on possible improvements of the relative frequency estimates by some smoothing techniques. It is today common practice to use additional feature functions like IBM-1 scores to obtain some kind of smoothing (Och et al., 2004; Koehn et al., 2003; Zens and Ney, 2004), but better estimation of the phrase probabilities is usually not addressed.

An alternative way to represent phrases is to define bilingual tuples. Smoothing, and context dependency, is obtained by using an n-gram model on these tuples. In this work, we have extended this approach by using a new smoothing technique that operates on a continuous representation of the tuples. Our method is distinguished by two characteristics: better estimation of the numerous unseen n-grams, and a **discriminative** estimation of the tuple probabilities. Results are provided on the BTEC task of the 2006 IWSLT evaluation for the translation direction Italian to English. This task provides very limited amount of resources in comparison to other tasks. Therefore, new techniques must be deployed to take the best advantage of the limited resources. We have chosen the Italian to English task because it is challenging to enhance a good quality translation task (over 40 BLEU percentage). Using the continuous space model for the **translation** and **target** language model, an improvement of 2.5 BLEU on the development data and 1.5 BLEU on the test data was observed.

Despite these encouraging results, we believe that additional research on improved estimation of probabilities in N-gram- or phrase-based statistical machine translation systems is needed. In particular, the problem of **generalization** to new translations seems to be promising to us. This could be addressed by the so-called factored phrase-based model as implemented in the Moses decoder (Koehn et al., 2007). In this approach words are decomposed into several factors. These factors are translated and a target phrase is generated. This model could be complemented by a factored continuous tuple N-gram. Factored word language models were already successfully used in speech recognition (Bilmes and Kirchhoff, 2003; Alexandrescu and Kirchhoff, 2006) and an extension to machine translation seems to be promising.

The described smoothing method was explicitly developed to tackle the data sparseness problem in tasks like the BTEC corpus. It is well known from language modeling that careful smoothing is less important when large amounts of data are available. We plan to investigate whether this also holds for smoothing of the probabilities in phrase- or tuple-based statistical machine translation systems.

6 Acknowledgments

This work has been partially funded by the European Union under the integrated project TC-STAR (IST-2002-FP6-506738), by the French Government under the project INSTAR (ANR JCJC06_143038) and the the Spanish government under a FPU grant and the project AVIVAVOZ (TEC2006-13964-C03).

References

A. Alexandrescu and K. Kirchhoff. 2006. Factored neural language models. In *HLT-NAACL*.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(2):1137–1155.

F. Vanden Berghen and H. Bersini. 2005. CONDOR, a new parallel, constrained extension of powell's UOBYQA algorithm: Experimental results and comparison with the DFO algorithm. *Journal of Computational and Applied Mathematics*, 181:157–175.

J. A. Bilmes and K. Kirchhoff. 2003. Factored language models and generalized backoff. In *HLT-NAACL*.

P. Brown, S. Della Pietra, V. J. Della Pietra, and R: Mercer. 1993. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–311.

F. Casacuberta, D. Llorens, C. Martínez, S. Molau, F. Nevado, H. Ney, M. Pastor, D. Picó, A. Sanchis, E. Vidal, and J.M. Vilar. 2001. Speech-to-speech translation based on finite-state transducers. *International Conference on Acoustic, Speech and Signal Processing*, 1.

S. F. Chen and J. T. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *CSL*, 13(4):359–394.

G. Foster, R. Kuhn, and H. Johnson. 2006. Phrasetable smoothing for statistical machine translation. In *EMNLP06*, pages 53–61.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrased-based machine translation. In *Human Language Technology Conference (HLT-NAACL)*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, demonstration session*.

J.B. Mariño, R.E. Banchs, J.M. Crego, A. de Gispert, P. Lambert, J.A.R. Fonollosa, and M. R. Costa-jussà. 2006. Bilingual n-gram statistical machine translation. *Computational Linguistics*, 32(4):527–549, December.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*, pages 295–302.

F. J. Och, C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine translation. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Copora*, pages 20–28.

F.-J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*, pages 161–168.

M. Paul. 2006. Overview of the IWSLT 2006 campaign. In *IWSLT*, pages 1–15.

H. Schwenk, M. R. Costa-jussà, and J. A. R. Fonollosa. 2006. Continuous space language models for the iwslt 2006 task. *IWSLT*, pages 166–173.

H. Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21:492–518.

A. Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP*, pages II: 901–904.

T. Takezawa, E. Sumita, F. Sugaya, H. Yamamoto, and S. Yamamoto. 2002. Toward a borad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *LREC*, pages 147–152.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT/NACL*, pages 257–264.

Morphological Disambiguation of Hebrew: A Case Study in Classifier Combination

Danny Shacham
Department of Computer Science
University of Haifa
Haifa, Israel
dannysh@gmail.com

Shuly Wintner
Department of Computer Science
University of Haifa
Haifa, Israel
shuly@cs.haifa.ac.il

Abstract

Morphological analysis and disambiguation are crucial stages in a variety of natural language processing applications, especially when languages with complex morphology are concerned. We present a system which disambiguates the output of a morphological analyzer for Hebrew. It consists of several simple classifiers and a module which combines them under linguistically motivated constraints. We investigate a number of techniques for combining the predictions of the classifiers. Our best result, 91.44% accuracy, reflects a 25% reduction in error rate compared with the previous state of the art.

1 Introduction

Morphological analysis and disambiguation are crucial pre-processing steps for a variety of natural language processing applications, from search and information extraction to machine translation. For languages with complex morphology these are non-trivial processes. This paper presents a morphological disambiguation module for Hebrew which uses a sophisticated combination of classifiers to rank the analyses produced by a morphological analyzer. This work has a twofold contribution: first, our system achieves over 91% accuracy on the full disambiguation task, reducing the error rate of the previous state of the art by 25%. More generally, we explore several ways for combining the predictions of simple classifiers under constraints; the insight gained from these experiments will be useful for other applications of machine learning to complex (morphological and other) problems.

In the remainder of this section we discuss the complexity of Hebrew morphology, the challenge of morphological disambiguation and related work. We describe our methodology in Section 2: we use basic, naïve classifiers (Section 3) to predict some components of the analysis, and then combine them in several ways (Section 4) to predict a consistent result. We analyze the errors of the system in Section 5 and conclude with suggestions for future work.

1.1 Linguistic background

Hebrew morphology is rich and complex.[1] The major word formation machinery is root-and-pattern, and inflectional morphology is highly productive and consists of prefixes, suffixes and circumfixes. Nouns, adjectives and numerals inflect for number (singular, plural and, in rare cases, also dual) and gender (masculine or feminine). In addition, all these three types of nominals have two phonologically and morphologically distinct forms, known as the *absolute* and *construct* states. In the standard orthography approximately half of the nominals appear to have identical forms in both states, a fact which substantially increases the ambiguity. In addition, nominals take possessive pronominal suffixes which inflect for number, gender and person.

Verbs inflect for number, gender and person (first, second and third) and also for a combination of tense and aspect/mood, referred to simply as ‘tense’ below. Verbs can also take pronominal suffixes, which are interpreted as direct objects, and in some cases can also take nominative pronominal suffixes. A peculiarity of Hebrew verbs is that the participle form

[1] To facilitate readability we use a straight-forward transliteration of Hebrew using ASCII characters, where the characters (in Hebrew alphabetic order) are: abgdhwzxviklmnsypcqr$t.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 439–447, Prague, June 2007. ©2007 Association for Computational Linguistics

can be used as present tense, but also as a noun or an adjective.

These matters are complicated further due to two sources: first, the standard Hebrew orthography leaves most of the vowels unspecified. On top of that, the script dictates that many particles, including four of the most frequent prepositions, the definite article, the coordinating conjunction and some subordinating conjunctions, all attach to the words which immediately follow them. When the definite article *h* is prefixed by one of the prepositions *b*, *k* or *l*, it is assimilated with the preposition and the resulting form becomes ambiguous as to whether or not it is definite. For example, *bth* can be read either as *b+th* “in tea” or as *b+h+th* “in the tea”. Thus, the form *$bth* can be read as an inflected stem (the verb “capture”, third person singular feminine past), as *$+bth* “that+field”, *$+b+th* “that+in+tea”, *$+b+h+th* “that in the tea”, *$bt+h* “her sitting” or even as *$+bt+h* “that her daughter”.

An added complexity stems from the fact that there are two main standards for the Hebrew script: one in which vocalization diacritics, known as *niqqud* “dots”, decorate the words, and another in which the dots are missing, and other characters represent some, but not all of the vowels. Most of the texts in Hebrew are of the latter kind; unfortunately, different authors use different conventions for the undotted script. Thus, the same word can be written in more than one way, sometimes even within the same document. This fact adds significantly to the degree of ambiguity.

Our departure point in this work is HAMSAH (Yona and Wintner, 2007), a wide coverage, linguistically motivated morphological analyzer of Hebrew, which was recently re-implemented in Java and made available from the Knowledge Center for Processing Hebrew (`http://mila.cs.technion.ac.il/`). The output that HAMSAH produces for the form *$bth* is illustrated in Table 1. In general, it includes the part of speech (POS) as well as sub-category, where applicable, along with several POS-dependent features such as number, gender, tense, nominal state, definitness, etc.

1.2 The challenge of disambiguation

Identifying the correct morphological analysis of a given word in a given context is an important and non-trivial task. Unlike POS tagging, the task does not involve assigning an analysis to words which the analyzer does not recognize. However, selecting an analysis immediately induces a POS tagging for the target word (by projecting the analysis on the POS coordinate). Our main contribution in this work is a system that solves this problem with high accuracy.

Compared with POS tagging of English, morphological disambiguation of Hebrew is a much more complex endeavor due to the following factors:

Segmentation A single token in Hebrew can actually be a sequence of more than one lexical item. For example, analysis 4 of Table 1 (*$+b+h+th* “that+in+the+tea”) corresponds to the tag sequence IN+IN+DT+NN.

Large tagset The number of different tags in a language such as Hebrew (where the POS, morphological features and prefix and suffix particles are considered) is huge. HAMSAH produces 22 different parts of speech, some with subcategories; 6 values for the number feature (including disjunctions of values), 4 for gender, 5 for person, 7 for tense and 3 for nominal state. Possessive pronominal suffixes can have 15 different values, and prefix particle sequences can theoretically have hundreds of different forms. While not all the combinations of these values are possible, we estimate the number of possible analyses to be in the thousands.

Ambiguity Hebrew is highly ambiguous: HAMSAH outputs on average approximately 2.64 analyses per word token. Oftentimes two or more alternative analyses share the same part of speech, and in some cases two or more analyses are completely identical, except for their lexeme (see analyses 7 and 8 in Table 1). Morphological disambiguation of Hebrew is hence closer to the problem of word sense disambiguation than to standard POS tagging.

Anchors, which are often function words, are almost always morphologically ambiguous in Hebrew. These include most of the high-frequency forms. Many of the function words which help boost the performance of English POS tagging are actually prefix particles which add to the ambiguity in Hebrew.

#	Lexical ID	lexeme	POS	Num	Gen	Per	Ten	Stat	Def	Pref	Suf
1	17280	*$bt*	noun	sing	fem	N/A	N/A	abs	no		h
2	1379	*bt*	noun	sing	fem	N/A	N/A	abs	no	*$*	h
3	19130	*bth*	noun	sing	fem	N/A	N/A	abs	no	*$*	
4	19804	*th*	noun	sing	masc	N/A	N/A	abs	yes	*$+b+h*	
5	19804	*th*	noun	sing	masc	N/A	N/A	abs	no	*$+b*	
6	19804	*th*	noun	sing	masc	N/A	N/A	cons	no	*$+b*	
7	1541	*$bh*	verb	sing	fem	3	past	N/A	N/A		
8	9430	*$bt*	verb	sing	fem	3	past	N/A	N/A		

Table 1: The analyses of the form *$bth*

Word order in Hebrew is freer than in English.

1.3 Related work

The idea of using short context for morphological disambiguation dates back to Choueka and Lusignan (1985). Levinger et al. (1995) were the first to apply it to Hebrew, but their work was hampered by the lack of annotated corpora for training and evaluation. The first work which uses stochastic contextual information for morphological disambiguation in Hebrew is Segal (1999): texts are analyzed using the morphological analyzer of Segal (1997); then, each word in a text is assigned its most likely analysis, defined by probabilities computed from a small tagged corpus. In the next phase the system corrects its own decisions by using short context (one word to the left and one to the right of the target word). The corrections are also automatically learned from the tagged corpus (using transformation-based learning). In the last phase, the analysis is corrected by the results of a syntactic analysis of the sentence. The reported results are excellent: 96.2% accuracy. More reliable tests, however, reveal accuracy of 85.5% only (Lemberski, 2003, page 85). Furthermore, the performance of the program is unacceptable (the reported running time on "two papers" is thirty minutes).

Bar-Haim et al. (2005) use Hidden Markov Models (HMMs) to implement a segmenter and a tagger for Hebrew. The main innovation of this work is that it models word-segments (morphemes: prefixes, stem and suffixes), rather than full words. The accuracy of this system is 90.51% for POS tagging (a tagset of 21 POS tags is used) and 96.74% for segmentation (which is defined as identifying all prefixes, including a possibly assimilated definite article). As noted above, POS tagging does not amount to full morphological disambiguation.

Recently, Adler and Elhadad (2006) presented an unsupervised, HMM-based model for Hebrew morphological disambiguation, using a morphological analyzer as the only resource. A morpheme-based model learns both segmentation and tagging in parallel from a large (6M words) un-annotated corpus. Reported results are 92.32% for POS tagging and 88.5% for full morphological disambiguation. We refer to this result as the state of the art and use the same data for evaluation.

A supervised approach to morphological disambiguation of *Arabic* is given by Habash and Rambow (2005), who use two corpora of 120K words each to train several classifiers. Each morphological feature is predicted separately and then combined into a full disambiguation result. The accuracy of the disambiguator is 94.8%-96.2% (depending on the test corpus). Note, however, the high baseline of each classifier (96.6%-99.9%, depending on the classifier) and the full disambiguation task (87.3%-92.1%, depending on the corpus). We use a very similar approach below, but we experiment with more sophisticated methods for combining simple classifiers to induce a coherent prediction.

2 Methodology

For training and evaluation, we use a corpus of approximately 90,000 word tokens, consisting of newspaper texts, which was automatically analyzed using HAMSAH and then manually annotated (Elhadad et al., 2005). Annotation consists simply of selecting the correct analysis produced by the analyzer, or an indication that no such analysis exists. When the analyzer does not produce the correct analysis, it is added manually. This is the exact setup of the experiments reported by Adler and Elhadad (2006).

Table 2 lists some statistics of the corpus, and a histogram of analyses is given in Table 3. Table 4 lists the distribution of POS in the corpus.

Tokens	89347
Types	23947
Tokens with no correct analysis	8218
Tokens with no analysis	130
Degree of ambiguity	2.64

Table 2: Statistics of training corpus

# analyses	# tokens	# analyses	# tokens
1	38468	7	1977
2	15480	8	1309
3	11194	9	785
4	9934	10	622
5	5341	11	238
6	3472	>12	397

Table 3: Histogram of analyses

In all the experiments described in this paper we use SNoW (Roth, 1998) as the learning environment, with *winnow* as the update rule (using *perceptron* yielded very similar results). SNoW is a multi-class classifier that is specifically tailored for learning in domains in which the potential number of information sources (features) taking part in decisions is very large, of which NLP is a principal example. It works by learning a sparse network of linear functions over the feature space. SNoW has already been used successfully as the learning vehicle in a large collection of natural language related tasks and compared favorably with other classifiers (Punyakanok and Roth, 2001; Florian, 2002). Typically, SNoW is used as a classifier, and predicts using a winner-take-all mechanism over the activation values of the target classes. However, in addition to the prediction, it provides a reliable confidence level in the prediction, which enables its use in an inference algorithm that combines predictors to produce a coherent inference.

POS	# tokens	% tokens
Noun	25836	28.92
Punctuation	13793	15.44
Proper Noun	7238	8.10
Verb	7192	8.05
Preposition	7164	8.02
Adjective	5855	6.55
Participle	3213	3.60
Pronoun	2688	3.01
Adverb	2226	2.49
Conjunction	2021	2.26
Numeral	1972	2.21
Quantifier	951	1.06
Negation	848	0.95
Interrogative	80	0.09
Prefix	29	0.03
Interjection	12	0.01
Foreign	6	0.01
Modal	5	0.01

Table 4: POS frequencies

Following Daya et al. (2004) and Habash and Rambow (2005), we approach the problem of morphological disambiguation as a complex classification task. We train a classifier for each of the attributes that can contribute to the disambiguation of the analyses produced by HAMSAH (e.g., POS, tense, state). Each classifier predicts a small set of possible values and hence can be highly accurate. In particular, the basic classifiers do not suffer from problems of data sparseness. Of course, each simple classifier cannot fully disambiguate the output of HAMSAH, but it does induce a ranking on the analyses (see Table 6 below for the level of ambiguity which remains after each simple classifier is applied). Then, we combine the outcomes of the simple classifiers to produce a consistent ranking which induces a linear order on the analyses.

For evaluation we consider only the words that have at least one correct analysis in the annotated corpus. *Accuracy* is defined as the ratio between the number of words classified correctly and the total number of words in the test corpus that have a correct analysis. The *remaining level of ambiguity* is defined as the average number of analyses per word whose score is equal to the score of the top ranked analysis. This is greater than 1 only for the simple classifiers, where more than one analysis can have the same tag. In all the experiments we perform 10-fold cross-validation runs and report the average of the 10 runs, both on the entire corpus and on a subset of the corpus in which we only test on words which do *not* occur in the training corpus.

The *baseline* tag of the token w_i is the most prominent tag of all the occurrences of w_i in the corpus. The baseline for the combination is the most prominent analysis of all the occurrences of w_i in the corpus. If w_i does not occur in the corpus, we back off and select the most prominent tag in the corpus independently of the word w_i. For the combination baseline, we select the analysis of the most prominent lexical ID, chosen from the list of all possible lexical IDs of w_i. If there is more than one possible value, one top-ranking value is chosen at random.

3 Basic Classifiers

The simple classifiers are all built in the same way. They are trained on feature vectors that are generated from the output of the morphological analyzer, and tested on a clean output of the same analyzer. We defined several classifiers for the attributes of the morphological analyses. Since some attributes do not apply to all the analyses, we add a value of 'N/A' for the inapplicable attributes. An annotated corpus was needed in all those classifiers for training. We list the basic classifiers below.

POS 22 values (only 18 in our corpus), see Table 4.

Gender 'Masculine', 'Feminine', 'Masculine and feminine', 'N/A'.

Number 'Singular', 'Plural', 'Dual', 'N/A'.

Person 'First', 'Second', 'Third', 'N/A'.

Tense 'Past', 'Present', 'Participle', 'Future', 'Imperative', 'Infinitive', 'Bare Infinitive', 'N/A'.

Definite Article 'Def', 'indef', 'N/A'. Identifies also implicit (assimilated) definiteness.

Status 'Absolute', 'Construct' and 'N/A'.

Segmentation Predicts the number of letters which are prefix particles. Possible values are [0-6], 6 being the length of longest possible prefix sequence. Does not identify implicit definiteness.

Has properties A binary classifier which distinguishes between atomic POS categories (e.g., conjunction or negation) and categories whose words have attributes (such as nouns or verbs).

Each word in the training corpus induces features that are generated for itself and its immediate neighbors, using the output of the morphological analyzer. For each word in the window, we generate the following features: POS, number, gender, person, tense, state, definiteness, prefixes (where each possible prefix is a binary feature), suffix (binary: is there word suffixed?), number/gender/person of suffix, surface form, lemma, conjunction of the surface form and the POS, conjunction of the POS and the POS of prefixes and suffixes, and some disjunctions of POS. The total number of features for each example is huge (millions), but feature vectors are very sparse.

The simple classifiers can be configured in several ways. First, the size of the window around the target word had to be determined, and we experimented with several sizes, up to ± 3 words. Another issue is feature generation. It is straight-forward during training, but during evaluation and testing the feature extractor is presented only with the set of analyses produced by HAMSAH for each word, and has no access to the *correct* analysis. We experimented with two methods for tackling this problem: produce the *union* of all possible values for each feature; or select a single analysis, the baseline one, for each word, and generate only the features induced by this analysis. While this problem is manifested only during testing, it impacts also the training procedure, and so we experimented with feature generation at training using the correct analysis, the union of the analyses or the baseline analysis. The results of the experiments for the POS classifier are shown in Table 5. The best configuration uses a window of two words before and one word after the target word. For both testing and training we generate features using the baseline analysis.

With this setup, the accuracy of all the classifiers is shown in Table 6. We report results on two tasks: the entire test corpus; and words in the test corpus which do not occur in the training corpus, a much harder task. We list the *accuracy*, remaining level of *ambiguity* and reduction in error rate *ERR*, com-

Training	Testing	1 - 2	2 - 1	2 - 2	1 - 3	3 - 1	2 - 3	3 - 2	3 - 3
correct	baseline	91.37	91.53	91.69	91.55	91.69	91.83	91.75	**92.01**
correct	all	79.15	79.55	80.53	80.07	80.13	80.75	81.00	**82.07**
all	baseline	93.41	93.38	93.22	93.42	93.53	93.59	93.51	**93.61**
all	all	93.37	93.42	93.28	93.2	**93.61**	93.05	93.48	93.15
baseline	baseline	94.93	**94.97**	94.8	94.86	94.84	94.72	94.67	94.61
baseline	all	84.48	84.78	84.82	**85.65**	84.97	85.13	85.03	85.45

Table 5: Architectural configurations of the POS classifier: columns reflect the window size, rows refer to training and testing feature generation

	All words				Unseen words		
	baseline	classifier			baseline	classifier	
	accuracy	accuracy	ambiguity	ERR	accuracy	accuracy	ERR
POS	93.01	94.97	1.46	28.04	84.67	88.65	25.96
Gender	96.34	96.74	1.86	10.93	92.15	94.38	28.41
Number	96.79	97.92	1.91	35.20	92.35	95.91	46.54
Person	98.14	98.62	2.25	25.81	94.04	96.50	41.28
Tense	98.40	98.69	2.21	18.12	94.80	96.37	30.19
Definite Article	93.90	95.76	1.83	30.49	85.38	91.77	43.71
Status	92.73	95.06	1.57	32.05	84.46	89.85	34.68
Segmentation	99.12	97.80	2.25	—	97.67	97.66	—
Has properties	97.63	98.11	2.26	20.25	95.91	95.97	1.47

Table 6: Accuracy of the simple classifiers: ERR is reduction in error rate, compared with the baseline

pared with the baseline.

4 Combination of Classifiers

Given a set of simple classifiers, we now investigate various ways for combining their predictions. These predictions may be contradicting (for example, the POS classifier can predict ‘noun’ while the tense classifier predicts ‘past’), and we use the constraints imposed by the morphological analyzer to enforce a consistent analysis.

First, we define a naïve combination along the lines of Habash and Rambow (2005). The scores assigned by the simple classifiers (except segmentation, for which we use the baseline) to each analysis are accumulated, and the score of the complete analysis is their sum (experiments with different weights to the various classifiers proved futile). Even after the combination, the remaining level of ambiguity is 1.05; in ambiguous cases back off to the baseline analysis, and then choose at random one of the top-ranking analyses. The result of the combination is shown in Table 7.

	baseline	classifier	ERR
All words	86.11	**90.26**	29.88
Unseen words	67.53	78.52	33.85

Table 7: Results of the naïve combination

Next, we define a hierarchical combination in which we try to incorporate more linguistic knowledge pertaining to the dependencies between the classifiers. As a pre-processing step we classify the target word to one of two groups, using the *has properties* classifier. Then, we predict the main POS of the target word, and take this prediction to be true; we then apply only the subset of the other classifiers that are relevant to the main POS.

The results of the hierarchical combination are shown in Table 8. As can be seen, the hierarchical combination performs worse than the naïve one. We conjecture that this is because the hierarchical combination does not fully disambiguate, and a random top-ranking analysis is chosen more often than in the case of the naïve combination.

	naïve	hierarchical	ERR
All words	90.26	89.61	—
Unseen words	78.52	78.08	—

Table 8: Results of the hierarchial combination

The combination of independent classifiers under the constraints imposed by the possible morphological analyses is intended to capture context-dependent constraints on possible sequences of analyses. Such constraints are stochastic in nature, but linguistic theory tells us that several *hard* (deterministic) constraints also exist which rule out certain sequences of otherwise possible analyses. We now explore the utility of implementing such constraints to filter out linguistically impossible sequences.

Using several linguistic sources, we defined a set of constraints, each of which is a linguistically impossible sequence of analyses (all sequences are of length 2, although in principle longer ones could have been defined). We then checked the annotated corpus for violations of these constraints; we used the corpus to either verify the correctness of a constraint or further refine it (or abandon it altogether, in some cases). We then re-iterated the process with the new set of constraints.

The result was a small set of six constraints which are not violated in our annotated corpus. We used the constraints to rule out some of the paths defined by the possible outcomes of the morphological analyzer on a sequence of words. Each of the constraints below contributes a non-zero reduction in the error rate of the disambiguation module.The (slightly simplified) constraints are:

1. A verb in any tense but present cannot be followed by the genitive preposition '$l' (of).

2. A preposition with no attached pronomial suffix must be followed by a nominal phrase. This rule is relaxed for some prepositions which can be followed by the prefix '$'.

3. The preposition 'at' must be followed by a definite nominal phrase.

4. Construct-state words must be followed by a nominal phrase.

5. A sequence of two verbs is only allowed if: one of them is the verb 'hih' (be); one of them has a prefix; the second is infinitival; or the first is imperative and the second is in future tense.

6. A non-numeral quantifier must be followed by either a nominal phrase or a punctuation.

Imposing the linguistically motivated constraints on the classifier combination improved the results to some extent, as depicted in Table 9. The best results are obtained when the constraints are applied to the hierarchical combination.

5 Error analysis

We conducted extensive error analysis of both the simple classifiers and the combination module. The analysis was performed over one fold of the annotated corpus (8933 tokens). Table 10 depicts, for some classifiers, a subset of the confusion matrix: it lists the *correct* tag, the *chosen*, or predicted, tag, the number of occurrences of the specific error and the total number of errors made by the classifier.

classifier	correct	chosen	#	total
has props	yes	no	110	167
	no	yes	57	
segmentation	1	0	160	176
state	const	abs	154	412
definiteness	def	indef	98	300

Table 10: Simple classifiers, confusion matrix

Several patterns can be observed in Table 10. The 'has properties' classifier is biased towards predicting 'yes' instead of 'no'. The 'segmentation' classifier, which predicts the length of the prefix, also displays a clear bias. In almost 90% of its errors it predicts no prefix instead of a prefix of length one. 'Status' and 'definiteness' are among the weakest classifiers, biased towards the default.

Other classifiers make more sporadic types of errors. Of particular interest is the POS classifier. Here, when adjectives are mis-predicted, they are predicted as nouns. This can be explained by the morphological similarity of the two categories, and in particular by the similar syntactic contexts in which they occur. Similarly, almost 90% of mis-predicted verbs are predicted to be either nouns

	naïve	naïve + consts	ERR	hier.	hier. + cons	ERR
All words	90.26	90.90	6.57	89.61	**91.44**	17.61
Unseen words	78.52	79.56	4.84	78.08	81.74	16.70

Table 9: Accuracy results of various combination architectures. *ERR* is reduction in error rate due to the hard constraints. The best results are obtained using the hierarchical combination with hard constraints.

or adjectives, probably resulting from present-tense verbs in the training corpus which, in Hebrew, have similar distribution to nouns and adjectives.

The analysis of errors in the combination is more interesting. On the entire corpus, the disambiguator makes 7927 errors. Of those, 1476 (19%) are errors in which the correct analysis differs from the chosen one *only* in the value of the 'state' feature. Furthermore, in 1341 of the errors (17%) the system picks the correct analysis up to the value of 'definiteness'; of those, 1275 (16% of the errors) are words in which the definite article is assimilated in a preposition. In sum, many of the errors seem to be in the real tough cases.

6 Conclusions

Morphological disambiguation of Hebrew is a difficult task which involves, in theory, thousands of possible tags. We reconfirm the results of Daya et al. (2004) and Habash and Rambow (2005), which show that decoupling complex morphological tasks into several simple tasks improves the accuracy of classification. Our best result, 91.44% accuracy, reflects a reduction of 25% in error rate compared to the previous state of the art (Adler and Elhadad, 2006), and almost 40% compared to the baseline. We also show that imposing few context-dependent constraints on possible sequences of analyses improves the accuracy of the disambiguation. The disambiguation module will be made available through the Knowledge Center for Processing Hebrew (`http://mila.cs.technion.ac.il/`).

We believe that these results can be further improved in various ways. The basic classifiers can benefit from more detailed feature engineering and careful tuning of the parameters of the learning environment. There are various ways in which interrelated classifiers can be combined; we only explored three here. Using other techniques, such as inference-based training, in which the feature generation for training is done step by step, using information inferred in the previous step, is likely to yield better accuracy. We also believe that further linguistic exploration, based on deeper error analysis, will result in more hard constraints which can reduce the error rate of the combination module. Finally, we are puzzled by the differences between Hebrew and Arabic (for which the baseline and the current state of the art are significantly higher) on this task. We intend to investigate the linguistic sources for this puzzle in the future.

Acknowledgements

We are extremely grateful to Dan Roth for his continuing support and advise; to Meni Adler for providing the annotated corpus; to Dalia Bojan and Alon Itai for the implementation of the morphological analyzer; to Yariv Louck for the implementation of the deterministic constraints; to Nurit Melnik for help with error analysis; and to Yuval Nardi his help with statistical analysis. Thanks are due to Ido Dagan, Alon Lavie and Michael Elhadad for useful comments and advise. This research was supported by THE ISRAEL SCIENCE FOUNDATION (grant No. 137/06); by the Israel Internet Association; by the Knowledge Center for Processing Hebrew; and by the Caesarea Rothschild Institute for Interdisciplinary Application of Computer Science at the University of Haifa.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 665–672, Sydney, Australia, July. Association for Computational Linguistics.

Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. 2005. Choosing an optimal architecture for segmentation and

POS-tagging of Modern Hebrew. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 39–46, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Yaacov Choueka and Serge Lusignan. 1985. Disambiguation by short context. *Computers and the Humanities*, 19:147–157.

Ezra Daya, Dan Roth, and Shuly Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In *Proceedings of EMNLP'04*, pages 357–364, Barcelona, Spain, July.

Michael Elhadad, Yael Netzer, David Gabay, and Meni Adler. 2005. Hebrew morphological tagging guidelines. Technical report, Department of Computer Science, Ben Gurion University.

Radu Florian. 2002. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178. Taiwan.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Gennadiy Lemberski. 2003. Named entity recognition in Hebrew. Master's thesis, Department of Computer Science, Ben Gurion University, Beer Sheva, Israel, March. In Hebrew.

Moshe Levinger, Uzzi Ornan, and Alon Itai. 1995. Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 21(3):383–404, September.

Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems 13*, pages 995–1001. MIT Press.

Dan Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of AAAI-98 and IAAI-98*, pages 806–813, Madison, Wisconsin.

Erel Segal. 1997. Morphological analyzer for unvocalized Hebrew words. Unpublished work.

Erel Segal. 1999. Hebrew morphological analyzer for Hebrew undotted texts. Master's thesis, Technion, Israel Institute of Technology, Haifa, October. In Hebrew.

Shlomo Yona and Shuly Wintner. 2007. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*. To appear.

Enhancing Single-document Summarization by Combining RankNet and Third-party Sources

Krysta M. Svore
Microsoft Research
1 Microsoft Way
Redmond, WA 98052
ksvore@microsoft.com

Lucy Vanderwende
Microsoft Research
1 Microsoft Way
Redmond, WA 98052

Christopher J.C. Burges
Microsoft Research
1 Microsoft Way
Redmond, WA 98052

Abstract

We present a new approach to automatic summarization based on neural nets, called NetSum. We extract a set of features from each sentence that helps identify its importance in the document. We apply novel features based on news search query logs and Wikipedia entities. Using the RankNet learning algorithm, we train a pair-based sentence ranker to score every sentence in the document and identify the most important sentences. We apply our system to documents gathered from CNN.com, where each document includes highlights and an article. Our system significantly outperforms the standard baseline in the ROUGE-1 measure on over 70% of our document set.

1 Introduction

Automatic summarization was first studied almost 50 years ago by Luhn (Luhn, 1958) and has continued to be a steady subject of research. Automatic summarization refers to the creation of a shortened version of a document or cluster of documents by a machine, see (Mani, 2001) for details. The summary can be an abstraction or extraction. In an abstract summary, content from the original document may be paraphrased or generated, whereas in an extract summary, the content is preserved in its original form, i.e., sentences. Both summary types can involve sentence compression, but abstracts tend to be more condensed. In this paper, we focus on producing fully automated single-document extract summaries of newswire articles.

To create an extract, most automatic systems use linguistic and/or statistical methods to identify key words, phrases, and concepts in a sentence or across single or multiple documents. Each sentence is then assigned a score indicating the strength of presence of key words, phrases, and so on. Sentence scoring methods utilize both purely statistical and purely semantic features, for example as in (Vanderwende et al., 2006; Nenkova et al., 2006; Yih et al., 2007).

Recently, machine learning techniques have been successfully applied to summarization. The methods include binary classifiers (Kupiec et al., 1995), Markov models (Conroy et al., 2004), Bayesian methods (Daumé III and Marcu, 2005; Aone et al., 1998), and heuristic methods to determine feature weights (Schiffman, 2002; Lin and Hovy, 2002). Graph-based methods have also been employed (Erkan and Radev, 2004a; Erkan and Radev, 2004b; Mihalcea, 2005; Mihalcea and Tarau, 2005; Mihalcea and Radev, 2006).

In 2001–02, the Document Understanding Conference (DUC, 2001), issued the task of creating a 100-word summary of a single news article. The best performing systems (Hirao et al., 2002; Lal and Ruger, 2002) used various learning and semantic-based methods, although no system could outperform the baseline with statistical significance (Nenkova, 2005). After 2002, the single-document summarization task was dropped.

In recent years, there has been a decline in studies on automatic single-document summarization, in part because the DUC task was dropped, and in part because the task of single-document extracts may be counterintuitively more difficult than multi-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 448–457, Prague, June 2007. ©2007 Association for Computational Linguistics

document summarization (Nenkova, 2005). However, with the ever-growing internet and increased information access, we believe single-document summarization is essential to improve quick access to large quantities of information. Recently, CNN.com (CNN.com, 2007a) added “Story Highlights” to many news articles on its site to allow readers to quickly gather information on stories. These highlights give a brief overview of the article and appear as 3–4 related sentences in the form of bullet points rather than a summary paragraph, making them even easier to quickly scan.

Our work is motivated by both the addition of highlights to an extremely visible and reputable online news source, as well as the inability of past single-document summarization systems to outperform the extremely strong baseline of choosing the first n sentences of a newswire article as the summary (Nenkova, 2005). Although some recent systems indicate an improvement over the baseline (Mihalcea, 2005; Mihalcea and Tarau, 2005), statistical significance has not been shown. We show that by using a neural network ranking algorithm and third-party datasets to enhance sentence features, our system, NetSum, can outperform the baseline with statistical significance.

Our paper is organized as follows. Section 2 describes our two studies: summarization and highlight extraction. We describe our dataset in detail in Section 3. Our ranking system and feature vectors are outlined in Section 4. We present our evaluation measure in Section 5. Sections 6 and 7 report on our results on summarization and highlight extraction, respectively. We conclude in Section 8 and discuss future work in Section 9.

2 Our Task

In this paper, we focus on single-document summarization of newswire documents. Each document consists of three highlight sentences and the article text. Each highlight sentence is human-generated, but is based on the article. In Section 4 we discuss the process of matching a highlight to an article sentence. The output of our system consists of purely extracted sentences, where we do not perform any sentence compression or sentence generation. We leave such extensions for future work.

We develop two separate problems based on our document set. First, can we extract three sentences that best “match” the highlights as a whole? In this task, we concatenate the three sentences produced by our system into a single summary or *block*, and similarly concatenate the three highlight sentences into a single summary or *block*. We then compare our system’s block against the highlight block. Second, can we extract three sentences that best “match” the three highlights, such that ordering is preserved? In this task, we produce three sentences, where the first sentence is compared against the first highlight, the second sentence is compared against the second highlight, and the third sentence is compared against the third highlight. Credit is not given for producing three sentences that match the highlights, but are out of order. The second task considers ordering and compares sentences on an individual level, whereas the first task considers the three chosen sentences as a summary or block and disregards sentence order. In both tasks, we assume the title has been seen by the reader and will be listed above the highlights.

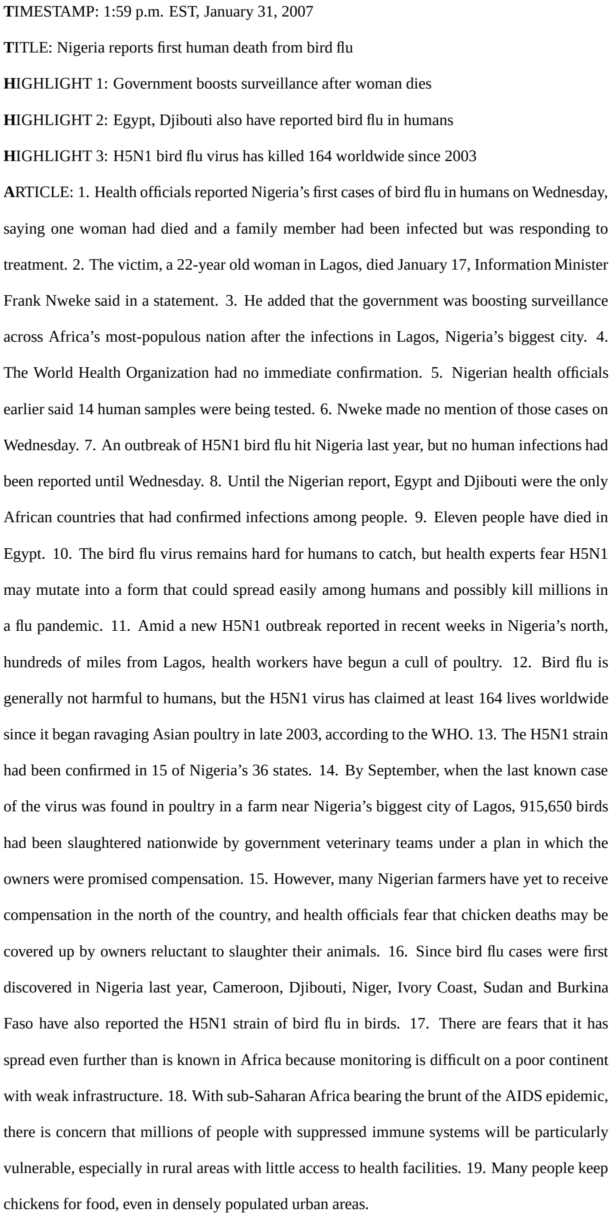
3 Evaluation Corpus

Our data consists of 1365 news documents gathered from CNN.com (CNN.com, 2007a). Each document was extracted by hand, where a maximum of 50 documents per day were collected. The documents were hand-collected on consecutive days during the month of February.

Each document includes the title, timestamp, story highlights, and article text. The timestamp on articles ranges from December 2006 to February 2007, since articles remain posted on CNN.com for up to several months. The story highlights are human-generated from the article text. The number of story highlights is between 3–4. Since all articles include at least 3 story highlights, we consider only the task of extracting three highlights from each article.

4 Description of Our System

Our goal is to extract three sentences from a single news document that best match various characteristics of the three document highlights. One way to identify the best sentences is to rank the sentences



TIMESTAMP: 1:59 p.m. EST, January 31, 2007

TITLE: Nigeria reports first human death from bird flu

HIGHLIGHT 1: Government boosts surveillance after woman dies

HIGHLIGHT 2: Egypt, Djibouti also have reported bird flu in humans

HIGHLIGHT 3: H5N1 bird flu virus has killed 164 worldwide since 2003

ARTICLE: 1. Health officials reported Nigeria's first cases of bird flu in humans on Wednesday, saying one woman had died and a family member had been infected but was responding to treatment. 2. The victim, a 22-year old woman in Lagos, died January 17, Information Minister Frank Nweke said in a statement. 3. He added that the government was boosting surveillance across Africa's most-populous nation after the infections in Lagos, Nigeria's biggest city. 4. The World Health Organization had no immediate confirmation. 5. Nigerian health officials earlier said 14 human samples were being tested. 6. Nweke made no mention of those cases on Wednesday. 7. An outbreak of H5N1 bird flu hit Nigeria last year, but no human infections had been reported until Wednesday. 8. Until the Nigerian report, Egypt and Djibouti were the only African countries that had confirmed infections among people. 9. Eleven people have died in Egypt. 10. The bird flu virus remains hard for humans to catch, but health experts fear H5N1 may mutate into a form that could spread easily among humans and possibly kill millions in a flu pandemic. 11. Amid a new H5N1 outbreak reported in recent weeks in Nigeria's north, hundreds of miles from Lagos, health workers have begun a cull of poultry. 12. Bird flu is generally not harmful to humans, but the H5N1 virus has claimed at least 164 lives worldwide since it began ravaging Asian poultry in late 2003, according to the WHO. 13. The H5N1 strain had been confirmed in 15 of Nigeria's 36 states. 14. By September, when the last known case of the virus was found in poultry in a farm near Nigeria's biggest city of Lagos, 915,650 birds had been slaughtered nationwide by government veterinary teams under a plan in which the owners were promised compensation. 15. However, many Nigerian farmers have yet to receive compensation in the north of the country, and health officials fear that chicken deaths may be covered up by owners reluctant to slaughter their animals. 16. Since bird flu cases were first discovered in Nigeria last year, Cameroon, Djibouti, Niger, Ivory Coast, Sudan and Burkina Faso have also reported the H5N1 strain of bird flu in birds. 17. There are fears that it has spread even further than is known in Africa because monitoring is difficult on a poor continent with weak infrastructure. 18. With sub-Saharan Africa bearing the brunt of the AIDS epidemic, there is concern that millions of people with suppressed immune systems will be particularly vulnerable, especially in rural areas with little access to health facilities. 19. Many people keep chickens for food, even in densely populated urban areas.

Figure 1: Example document containing highlights and article text. Sentences are numbered by their position. Article is from (CNN.com, 2007b).

using a machine learning approach, for example as in (Hirao et al., 2002). A train set is labeled such that the labels identify the best sentences. Then a set of features is extracted from each sentence in the train and test sets, and the train set is used to train the system. The system is then evaluated on the test set. The system learns from the train set the distribution of features for the best sentences and outputs a ranked list of sentences for each document. In this paper, we rank sentences using a neural network algorithm called RankNet (Burges et al., 2005).

4.1 RankNet

From the labels and features for each sentence, we train a model that, when run on a test set of sentences, can infer the proper ranking of sentences in a document based on information gathered during training about sentence characteristics. To accomplish the ranking, we use RankNet (Burges et al., 2005), a ranking algorithm based on neural networks.

RankNet is a pair-based neural network algorithm used to rank a set of inputs, in this case, the set of sentences in a given document. The system is trained on pairs of sentences (S_i, S_j), such that S_i should be ranked higher or equal to S_j. Pairs are generated between sentences in a single document, not across documents. Each pair is determined from the input labels. Since our sentences are labeled using ROUGE (see Section 4.3), if the ROUGE score of S_i is greater than the ROUGE score of S_j, then (S_i, S_j) is one input pair. The cost function for RankNet is the probabilistic cross-entropy cost function. Training is performed using a modified version of the back propagation algorithm for two layer nets (Le Cun et al., 1998), which is based on optimizing the cost function by gradient descent. A similar method of training on sentence pairs in the context of multi-document summarization was recently shown in (Toutanova et al., 2007).

Our system, NetSum, is a two-layer neural net trained using RankNet. To speed up the performance of RankNet, we implement RankNet in the framework of LambdaRank (Burges et al., 2006). For details, see (Burges et al., 2006; Burges et al., 2005). We experiment with between 5 and 15 hidden nodes and with an error rate between 10^{-2} and 10^{-7}.

We implement 4 versions of NetSum. The first version, NetSum(b), is trained for our first summarization problem (b indicates block). The pairs are generated using the maximum ROUGE scores l_1 (see Section 4.3). The other three rankers are trained to identify the sentence in the document that best matches highlight n. We train one ranker, NetSum(n), for each highlight n, for $n = 1, 2, 3$, resulting in three rankers. NetSum(n) is trained using pairs generated from the $l_{1,n}$ ROUGE scores between sentence S_i and highlight H_n (see Section 4.3).

4.2 Matching Extracted to Generated Sentences

In this section, we describe how to determine which sentence in the document best matches a given highlight. Choosing three sentences most similar to the three highlights is very challenging since the highlights include content that has been gathered across sentences and even paragraphs, and furthermore include vocabulary that may not be present in the text. Jing showed, for 300 news articles, that 19% of human-generated summary sentences contain no matching article sentence (Jing, 2002). In addition, only 42% of the summary sentences match the content of a single article sentence, where there are still semantic and syntactic transformations between the summary sentence and article sentence.. Since each highlight is human generated and does not exactly match any one sentence in the document, we must develop a method to identify how closely related a highlight is to a sentence. We use the ROUGE (Lin, 2004b) measure to score the similarity between an article sentence and a highlight sentence. We anticipate low ROUGE scores for both the baseline and NetSum due to the difficulty of finding a single sentence to match a highlight.

4.3 ROUGE

Recall-Oriented Understudy for Gisting Evaluation (Lin, 2004b), known as ROUGE, measures the quality of a model-generated summary or sentence by comparing it to a "gold-standard", typically human-generated, summary or sentence. It has been shown that ROUGE is very effective for measuring both single-document summaries and single-document headlines (Lin, 2004a).

ROUGE-N is a N-gram recall between a model-generated summary and a reference summary. We use ROUGE-N, for $N = 1$, for labeling and evaluation of our model-generated highlights.[1] ROUGE-1 and ROUGE-2 have been shown to be statistically similar to human evaluations and can be used with a single reference summary (Lin, 2004a). We have only one reference summary, the set of human-generated highlights, per document. In our work, the reference summary can be a single highlight sentence or the highlights as a block. We calculate ROUGE-N as

$$\frac{\sum_{gram_j \in R \cap S_i} Count(gram_j)}{\sum_{gram_j \in R} Count(gram_j)}, \tag{1}$$

where R is the reference summary, S_i is the model-generated summary, and N is the length of the N-gram $gram_j$.[2] The numerator cannot excede the number of N-grams (non-unique) in R.

We label each sentence S_i by its ROUGE-1 score. For the first problem of matching the highlights as a block, we label each S_i by l_1, the maximum ROUGE-1 score between S_i and each highlight H_n, for $n = 1, 2, 3$, given by $l_1 = \max_n(R(S_i, H_n))$.

For the second problem of matching three sentences to the three highlights individually, we label each sentence S_i by $l_{1,n}$, the ROUGE-1 score between S_i and H_n, given by $l_{1,n} = R(S_i, H_n)$. The ranker for highlight n, NetSum(n), is passed samples labeled using $l_{1,n}$.

4.4 Features

RankNet takes as input a set of samples, where each sample contains a label and feature vector. The labels were previously described in Section 4.3. In this section, we describe each feature in detail and motivate in part why each feature is chosen. We generate 10 features for each sentence S_i in each document, listed in Table 1. Each feature is chosen to identify characteristics of an article sentence that may match those of a highlight sentence. Some of the features such as position and N-gram frequencies are commonly used for scoring. Sentence scoring based on

[1] We use an implementation of ROUGE that does not perform stemming or stopword removal.

[2] ROUGE is typically used when the length of the reference summary is equal to length of the model-generated summary. Our reference summary and model-generated summary are different lengths, so there is a slight bias toward longer sentences.

Symbol	Feature Name
$F(S_i)$	Is First Sentence
$Pos(S_i)$	Sentence Position
$SB(S_i)$	SumBasic Score
$SB_b(S_i)$	SumBasic Bigram Score
$Sim(S_i)$	Title Similarity Score
$NT(S_i)$	Average News Query Term Score
$NT_+(S_i)$	News Query Term Sum Score
$NT_r(S_i)$	Relative News Query Term Score
$WE(S_i)$	Average Wikipedia Entity Score
$WE_+(S_i)$	Wikipedia Entity Sum Score

Table 1: Features used in our model.

sentence position, terms common with the title, appearance of keyword terms, and other cue phrases is known as the Edmundsonian Paradigm (Edmundson, 1969; Alfonesca and Rodriguez, 2003; Mani, 2001). We use variations on these features as well as a novel set of features based on third-party data.

Typically, news articles are written such that the first sentence summarizes the article. Thus, we include a binary feature $F(S_i)$ that equals 1 if S_i is the first sentence of the document: $F(S_i) = \delta_{i,1}$, where δ is the Kronecker delta function. This feature is used only for NetSum(b) and NetSum(1).

We include sentence position since we found in empirical studies that the sentence to best match highlight H_1 is on average 10% down the article, the sentence to best match H_2 is on average 20% down the article, and the sentence to best match H_3 is 31% down the article.[3] We calculate the position of S_i in document D as

$$Pos(S_i) = \frac{i}{\ell}, \tag{2}$$

where $i = \{1, \ldots, \ell\}$ is the sentence number and ℓ is the number of sentences in D.

We include the SumBasic score (Nenkova et al., 2006) of a sentence to estimate the importance of a sentence based on word frequency. We calculate the SumBasic score of S_i in document D as

$$SB(S_i) = \frac{\sum_{w \in S_i} p(w)}{|S_i|}, \tag{3}$$

where $p(w)$ is the probability of word w and $|S_i|$ is the number of words in sentence S_i. We calculate $p(w)$ as $p(w) = \frac{Count(w)}{|D|}$, where $Count(w)$ is the number of times word w appears in document D and $|D|$ is the number of words in document D. Note that the score of a sentence is the average probability of a word in the sentence.

We also include the SumBasic score over bigrams, where w in Eq 3 is replaced by bigrams and we normalize by the number of bigrams in S_i.

We compute the similarity of a sentence S_i in document D with the title T of D as the relative probability of title terms $t \in T$ in S_i as

$$Sim(S_i) = \frac{\sum_{t \in S_i} p(t)}{|S_i|}, \tag{4}$$

where $p(t) = \frac{Count(t)}{|T|}$ is the number of times term t appears in T over the number of terms in T.

The remaining features we use are based on third-party data sources. Previously, third-party sources such as WordNet (Fellbaum, 1998), the web (Jagalamudi et al., 2006), or click-through data (Sun et al., 2005) have been used as features. We propose using news query logs and Wikipedia entities to enhance features. We base several features on query terms frequently issued to Microsoft's news search engine http://search.live.com/news, and entities[4] found in the online open-source encyclopedia Wikipedia (Wikipedia.org, 2007). If a query term or Wikipedia entity appears frequently in a CNN document, then we assume highlights should include that term or entity since it is important on both the document and global level. Sentences containing query terms or Wikipedia entities therefore contain important content. We confirm the importance of these third-party features in Section 7.

We collected several hundred of the most frequently queried terms in February 2007 from the news query logs. We took the daily top 200 terms for 10 days. Our hypothesis is that a sentence with a higher number of news query terms should be a better candidate highlight. We calculate the average probability of news query terms q in S_i as

$$NT(S_i) = \frac{\sum_{q \in S_i} p(q)}{|q \in S_i|}, \tag{5}$$

[3] Though this is not always the case, as the sentence to match H_2 precedes that to match H_1 in 22.03% of documents, and the sentence to match H_3 precedes that to match H_2 in 29.32% of and precedes that to match H_1 in 28.81% of documents.

[4] We define an entity as a title of a Wikipedia page.

where $p(q)$ is the probability of a news term q and $|q \in S_i|$ is the number of news terms in S_i. $p(q) = \frac{Count(q)}{|q \in D|}$, where $Count(q)$ is the number of times term q appears in D and $|q \in D|$ is the number of news query terms in D.

We also include the sum of news query terms in S_i, given by $NT_+(S_i) = \sum_{q \in S_i} p(q)$, and the relative probability of news query terms in S_i, given by $NT_r(S_i) = \frac{\sum_{q \in S_i} p(q)}{|S_i|}$.

We perform term disambiguation on each document using an entity extractor (Cucerzan, 2007). Terms are disambiguated to a Wikipedia entity only if they match a surface form in Wikipedia. Wikipedia surface forms are terms that disambiguate to a Wikipedia entity and link to a Wikipedia page with the entity as its title. For example, "WHO" and "World Health Org." both refer to the World Health Organization, and should disambiguate to the entity "World Health Organization". Sentences in CNN document D that contain Wikipedia entities that frequently appear in CNN document D are considered important. We calculate the average Wikipedia entity score for S_i as

$$WE(S_i) = \frac{\sum_{e \in S_i} p(e)}{|e \in S_i|}, \tag{6}$$

where $p(e)$ is the probability of entity e, given by $p(e) = \frac{Count(e)}{|e \in D|}$, where $Count(e)$ is the number of times entity e appears in CNN document D and $|e \in D|$ is the total number of entities in CNN document D.

We also include the sum of Wikipedia entities, given by $WE_+(S_i) = \sum_{e \in S_i} p(e)$.

Note that all features except position features are a variant of SumBasic over different term sets. All features are computed over sentences where every word has been lowercased and punctuation has been removed after sentence breaking. We examined using stemming, but found stemming to be ineffective.

5 Evaluation

We evaluate the performance of NetSum using ROUGE and by comparing against a baseline system. For the first summarization task, we compare against the baseline of choosing the first three sentences as the block summary. For the second highlights task, we compare NetSum(n) against the baseline of choosing sentence n (to match highlight n). Both tasks are novel in attempting to match highlights rather than a human-generated summary.

We consider ROUGE-1 to be the measure of importance and thus train our model on ROUGE-1 (to optimize ROUGE-1 scores) and likewise evaluate our system on ROUGE-1. We list ROUGE-2 scores for completeness, but do not expect them to be substantially better than the baseline since we did not directly optimize for ROUGE-2.[5]

For every document in our corpus, we compare NetSum's output with the baseline output by computing ROUGE-1 and ROUGE-2 between the highlight block and NetSum and between the highlight block and the block of sentences. Similarly, for each highlight, we compute ROUGE-1 and ROUGE-2 between highlight n and NetSum(n) and between highlight n and sentence n, for $n = 1, 2, 3$. For each task, we calculate the average ROUGE-1 and ROUGE-2 scores of NetSum and of the baseline. We also report the percent of documents where the ROUGE-1 score of NetSum is equal to or better than the ROUGE-1 score of the baseline.

We perform all experiments using five-fold cross-validation on our dataset of 1365 documents. We divide our corpus into five random sets and train on three combined sets, validate on one set, and test on the remaining set. We repeat this procedure for every combination of train, validation, and test sets. Our results are the micro-averaged results on the five test sets. For all experiments, Table 3 lists the statistical tests performed and the significance of performance differences between NetSum and the baseline at 95% confidence.

6 Results: Summarization

We first find three sentences that, as a block, best match the three highlights as a block. NetSum(b) produces a ranked list of sentences for each document. We create a block from the top 3 ranked sentences. The baseline is the block of the first 3 sentences of the document. A similar baseline outper-

[5] NetSum can directly optimize for any measure by training on it, such as training on ROUGE-2 or on a weighted sum of ROUGE-1 and ROUGE-2 to optimize both. Thus, ROUGE-2 scores could be further improved. We leave such studies for future work.

System	Av. ROUGE-1	Av. ROUGE-2
Baseline	0.4642 ± 0.0084	0.1726 ± 0.0064
NetSum(b)	**0.4956** ± 0.0075	0.1775 ± 0.0066

Table 2: Results on summarization task with standard error at 95% confidence. Bold indicates significance under paired tests.

	ROUGE-1			ROUGE-2		
System	1	2	3	1	2	3
NetSum(b)	**x**	**x**	**x**	**x**	o	o
NetSum(1)	**x**	**x**	**x**	o	o	o
NetSum(2)	**x**	**x**	**x**	**x**	o	**x**
NetSum(3)	**x**	**x**	**x**	**x**	**x**	**x**

Table 3: Paired tests for statistical significance at 95% confidence between baseline and NetSum performance; 1: McNemar, 2: Paired t-test, 3: Wilcoxon signed-rank. "**x**" indicates pass, "o" indicates fail. Since our studies are pair-wise, tests listed here are more accurate than error bars reported in Tables 2–5.

forms all previous systems for news article summarization (Nenkova, 2005) and has been used in the DUC workshops (DUC, 2001).

For each block produced by NetSum(b) and the baseline, we compute the ROUGE-1 and ROUGE-2 scores of the block against the set of highlights as a block. For 73.26% of documents, NetSum(b) produces a block with a ROUGE-1 score that is equal to or better than the baseline score. The two systems produce blocks of equal ROUGE-1 score for 24.69% of documents. Under ROUGE-2, NetSum(b) performs equal to or better than the baseline on 73.19% of documents and equal to the baseline on 40.51% of documents.

Table 2 shows the average ROUGE-1 and ROUGE-2 scores obtained with NetSum(b) and the baseline. NetSum(b) produces a higher quality block on average for ROUGE-1.

Table 4 lists the sentences in the block produced by NetSum(b) and the baseline block, for the articles shown in Figure 1. The NetSum(b) summary achieves a ROUGE-1 score of 0.52, while the baseline summary scores only 0.36.

System	Sent. #	ROUGE-1
Baseline	S_1, S_2, S_3	0.36
NetSum(b)	S_1, S_7, S_{15}	**0.52**

Table 4: Block results for the block produced by NetSum(b) and the baseline block for the example article. ROUGE-1 scores computed against the highlights as a block are listed.

7 Results: Highlights

Our second task is to extract three sentences from a document that best match the three highlights in order. To accomplish this, we train NetSum(n) for each highlight $n = 1, 2, 3$. We compare NetSum(n) with the baseline of picking the nth sentence of the document. We perform five-fold cross-validation across our 1365 documents. Our results are reported for the micro-average of the test results. For each highlight n produced by both NetSum(n) and the baseline, we compute the ROUGE-1 and ROUGE-2 scores against the nth highlight.

We expect that beating the baseline for $n = 1$ is a more difficult task than for $n = 2$ or 3 since the first sentence of a news article typically acts as a summary of the article and since we expect the first highlight to summarize the article. NetSum(1), however, produces a sentence with a ROUGE-1 score that is equal to or better than the baseline score for 93.26% of documents. The two systems produce sentences of equal ROUGE-1 scores for 82.84% of documents. Under ROUGE-2, NetSum(1) performs equal to or better than the baseline on 94.21% of documents.

Table 5 shows the average ROUGE-1 and ROUGE-2 scores obtained with NetSum(1) and the baseline. NetSum(1) produces a higher quality sentence on average under ROUGE-1.

The content of highlights 2 and 3 is typically from later in the document, so we expect the baseline to not perform as well in these tasks. NetSum(2) outperforms the baseline since it is able to identify sentences from further down the document as important. For 77.73% of documents, NetSum(2) produces a sentence with a ROUGE-1 score that is equal to or better than the score for the baseline. The two systems produce sentences of equal ROUGE-1 score for 33.92% of documents. Under ROUGE-2, NetSum(2) performs equal to or better than the baseline

System	Av. ROUGE-1	Av. ROUGE-2
Baseline(1)	0.4343 ± 0.0138	0.1833 ± 0.0095
NetSum(1)	**0.4478** ± 0.0133	0.1857 ± 0.0085
Baseline(2)	0.2451 ± 0.0128	0.0814 ± 0.0106
NetSum(2)	**0.3036** ± 0.0117	**0.0877** ± 0.0107
Baseline(3)	0.1707 ± 0.0103	0.0412 ± 0.0069
NetSum(3)	**0.2603** ± 0.0133	**0.0615** ± 0.0075

Table 5: Results on ordered highlights task with standard error at 95% confidence. Bold indicates significance under paired tests.

System	Sent. #	ROUGE-1
Baseline	S_1	0.167
NetSum(1)	S_1	0.167
Baseline	S_2	0.111
NetSum(2)	S_1	**0.556**
Baseline	S_3	0.000
NetSum(3)	S_{15}	**0.400**

Table 6: Highlight results for highlight n produced by NetSum(n) and highlight n produced by the baseline for the example article. ROUGE-1 scores computed against highlight n are listed.

84.84% of the time. For 81.09% of documents, NetSum(3) produces a sentence with a ROUGE-1 score that is equal to or better than the score for the baseline. The two systems produce sentences of equal ROUGE-1 score for 28.45% of documents. Under ROUGE-2, NetSum(3) performs equal to or better than the baseline 89.91% of the time.

Table 5 shows the average ROUGE-1 and ROUGE-2 scores obtained for NetSum(2), NetSum(3), and the baseline. Both NetSum(2) and NetSum(3) produce a higher quality sentence on average under both measures.

Table 6 gives highlights produced by NetSum(n) and the highlights produced by the baseline, for the article shown in Figure 1. The NetSum(n) highlights produce ROUGE-1 scores equal to or higher than the baseline ROUGE-1 scores.

In feature ablation studies, we confirmed that the inclusion of news-based and Wikipedia-based features improves NetSum's peformance. For example, we removed all news-based and Wikipedia-based features in NetSum(3). The resulting performance moderately declined. Under ROUGE-1, the baseline produced a better highlight on 22.34% of documents, versus only 18.91% when using third-party features. Similarly, NetSum(3) produced a summary of equal or better ROUGE-1 score on only 77.66% of documents, compared to 81.09% of documents when using third-party features. In addition, the average ROUGE-1 score dropped to 0.2182 and the average ROUGE-2 score dropped to 0.0448. The performance of NetSum with third-party features over NetSum without third-party features is statistically significant at 95% confidence. However, NetSum still outperforms the baseline without third-party features, leading us to conclude that RankNet and simple position and term frequency features contribute the maximum performance gains, but increased ROUGE-1 and ROUGE-2 scores are a clear benefit of third-party features.

8 Conclusions

We have presented a novel approach to automatic single-document summarization based on neural networks, called NetSum. Our work is the first to use both neural networks for summarization and third-party datasets for features, using Wikipedia and news query logs. We have evaluated our system on two novel tasks: 1) producing a block of highlights and 2) producing three ordered highlight sentences. Our experiments were run on previously unstudied data gathered from CNN.com. Our system shows remarkable performance over the baseline of choosing the first n sentences of the document, where the performance difference is statistically significant under ROUGE-1.

9 Future Work

An immediate future direction is to further explore feature selection. We found third-party features beneficial to the performance of NetSum and such sources can be mined further. In addition, feature selection for each NetSum system could be performed separately since, for example, highlight 1 has different characteristics than highlight 2.

In our experiments, ROUGE scores are fairly low because a highlight rarely matches the content of a single sentence. To improve NetSum's performance, we must consider extracting content across sentence boundaries. Such work requires a system to produce abstract summaries. We hope to incorporate sentence simplification and sentence splicing and merging in a future version of NetSum.

Another future direction is the identification of "hard" and "easy" inputs. Although we report average ROUGE scores, such measures can be misleading since some highlights are simple to match and some are much more difficult. A better system evaluation measure would incorporate the difficulty of the input and weight reported results accordingly.

References

E. Alfonesca and P. Rodriguez. 2003. Description of the uam system for generating very short summaries at DUC–2003. In *DUC 2003: Document Understanding Conference, May 31–June 1, 2003, Edmonton, Canada*.

C. Aone, M. Okurowski, and J. Gorlinsky. 1998. Trainable scalable summarization using robust nlp and machine learning. In *Proceedings of the 17th COLING and 36th ACL*.

C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to Rank using Gradient Descent. In Luc De Raedt and Stefan Wrobel, editors, *ICML*, pages 89–96. ACM.

C.J.C. Burges, R. Ragno, and Q. Le. 2006. Learning to rank with nonsmooth cost functions. In *NIPS 2006: Neural Information Processing Systems, December 4-7, 2006, Vancouver, CA*.

CNN.com. 2007a. Cable news network. http://www.cnn.com/.

CNN.com. 2007b. Nigeria reports first human death from bird flu. http://edition.cnn.com/2007/WORLD/africa/01/31/nigeria.bird.flu.ap/index.html?eref=edition_world.

J. Conroy, J. Schlesinger, J. Goldstein, and D. O'Leary. 2004. Left-brain/right-brain multi-document summarization. In *DUC 2004: Document Understanding Workshop, May 6–7, 2004, Boston, MA, USA*.

S. Cucerzan. 2007. Large scale named entity disambiguation based on wikipedia data. In *EMNLP 2007: Empirical Methods in Natural Language Processing, June 28-30, 2007, Prague, Czech Republic*.

H. Daumé III and D. Marcu. 2005. Bayesian multi-document summarization at mse. In *Proceedings of MSE*.

DUC. 2001. Document understanding conferences. http://www-nlpir.nist.gov/projects/duc/index.html.

H.P. Edmundson. 1969. New methods in automatic extracting. *Journal for the Association of Computing Machinery*, 16:159–165.

G. Erkan and D. R. Radev. 2004a. Lexpagerank: Prestige in multi-document text summarization. In *EMNLP 2004: Empirical Methods in Natural Language Processing, 2004, Barcelona, Spain*.

G. Erkan and D. R. Radev. 2004b. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

T. Hirao, Y. Sasaki, H. Isozaki, and E. Maeda. 2002. Ntt's text summarization system for DUC–2002. In *DUC 2002: Workshop on Text Summarization, July 11–12, 2002, Philadelphia, PA, USA*.

J. Jagalamudi, P. Pingali, and V. Varma. 2006. Query independent sentence scoring approach to DUC 2006. In *DUC 2006: Document Understanding Conference, June 8–9, 2006, Brooklyn, NY, USA*.

H. Jing. 2002. Using hidden markov modeling to decompose human-written summaries. *Computational Linguistics*, 4(28):527–543.

J. Kupiec, J. Pererson, and F. Chen. 1995. A trainable document summarizer. *Research and Development in Information Retrieval*, pages 68–73.

P. Lal and S. Ruger. 2002. Extract-based summarization with simplification. In *DUC 2002: Workshop on Text Summarization, July 11–12, 2002, Philadelphia, PA, USA*.

Y. Le Cun, L. Bottou, G.B. Orr, and K.R. Müller. 1998. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag.

C.Y. Lin and E. Hovy. 2002. Automated multi-document summarization in neats. In *Proceedings of the Human Language Technology Conference (HLT2002)*.

C.Y. Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation — how many samples are enough? In *Proceedings of the NTCIR Workshop 4, June 2–4, 2004, Tokyo, Japan*.

C.Y. Lin. 2004b. Rouge: A package for automatic evaluation of summaries. In *WAS 2004: Proceedings of the Workshop on Text Summarization Branches Out, July 25–26, 2004, Barcelona, Spain*.

H. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

I. Mani. 2001. *Automatic Summarization*. John Benjamins Pub. Co.

R. Mihalcea and D. R. Radev, editors. 2006. *Textgraphs: Graph-based methods for NLP*. New York City, NY.

R. Mihalcea and P. Tarau. 2005. An algorithm for language independent single and multiple document summarization. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP), October, 2005, Korea*.

R. Mihalcea. 2005. Language independent extractive summarization. In *ACL 2005: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, June, 2005, Ann Arbor, MI, USA*.

A. Nenkova, L. Vanderwende, and K. McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In E. N. Efthimiadis, S. T. Dumais, D. Hawking, and K. Järvelin, editors, *SIGIR*, pages 573–580. ACM.

A. Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005), Pittsburgh, PA*.

B. Schiffman. 2002. Building a resource for evaluating the importance of sentences. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*.

J.T. Sun, D. Shen, H.J. Zeng, Q. Yang, Y. Lu, and Z. Chen. 2005. Web-page summarization using click-through data. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, editors, *SIGIR*. ACM.

K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. 2007. The pythy summarization system: Microsoft research at DUC2007. In *DUC 2007: Document Understanding Conference, April 26–27, 2007, Rochester, NY, USA*.

L. Vanderwende, H. Suzuki, and C. Brockett. 2006. Microsoft research at DUC2006: Task-focused summarization with sentence simplification. In *DUC 2006: Document Understanding Workshop, June 8–9, 2006, Brooklyn, NY, USA*.

Wikipedia.org. 2007. Wikipedia org. http://www.wikipedia.org.

W.T. Yih, J. Goodman, L. Vanderwende, and H. Suzuki. 2007. Multi-document summarization by maximizing informative content words. In *IJCAI 2007: 20th International Joint Conference on Artificial Intelligence, January, 2007*.

Automatic Identification of Important Segments and Expressions for Mining of Business-Oriented Conversations at Contact Centers

Hironori Takeuchi[*], **L Venkata Subramaniam**[†], **Tetsuya Nasukawa**[*], **and Shourya Roy**[†]

[*]IBM Research, Tokyo Research Laboratory
Shimotsuruma 1623-14, Yamato-shi
Kanagawa 2428502 Japan
{hironori, nasukawa}@jp.ibm.com

[†]IBM Research, India Research Laboratory
Institutional Area 4, Block-C, Vasant Kunj
New Delhi 110070 India
{lvsubram, rshourya}@in.ibm.com

Abstract

Textual records of business-oriented conversations between customers and agents need to be analyzed properly to acquire useful business insights that improve productivity. For such an analysis, it is critical to identify appropriate textual segments and expressions to focus on, especially when the textual data consists of complete transcripts, which are often lengthy and redundant. In this paper, we propose a method to identify important segments from the conversations by looking for changes in the accuracy of a categorizer designed to separate different business outcomes. We extract effective expressions from the important segments to define various viewpoints. In text mining a viewpoint defines the important associations between key entities and it is crucial that the correct viewpoints are identified. We show the effectiveness of the method by using real datasets from a car rental service center.

1 Introduction

"Contact center" is a general term for customer service centers, help desks, and information phone lines. Many companies operate contact centers to sell their products, handle customer issues, and address product-related and services-related issues. In contact centers, analysts try to get insights for improving business processes from stored customer contact data. Gigabytes of customer contact records are produced every day in the form of audio recordings of speech, transcripts, call summaries, email, etc. Though analysis by experts results in insights that are very deep and useful, such analysis usually covers only a very small (1-2%) fraction of the total call volume and yet requires significant workload. The demands for extracting trends and knowledge from the whole text data collection by using text mining technology, therefore, are increasing rapidly.

In order to acquire valuable knowledge through text mining, it is generally critical to identify important expressions to be monitored and compared within the textual data. For example, given a large collection of contact records at the contact center of a manufacturer, the analysis of expressions for products and expressions for problems often leads to business value by identifying specific problems in a specific product. If 30% of the contact records with expressions for a specific product such as "ABC" contain expressions about a specific trouble such as "cracked", while the expressions about the same trouble appear in only 5% of the contact records for similar products, then it should be a clue that the product "ABC" may actually have a crack-related problem. An effective way to facilitate this type of analysis is to register important expressions in a lexicon such as "ABC" and "cracked" as associated respectively with their categories such as "product" and "problem" so that the behavior of terms in the same category can be compared easily. It is actually one of the most important steps of text mining to identify such relevant expressions and their categories that can potentially lead to some valuable insights. A failure in this step often leads to a failure in the text mining. Also, it has been considered an artistic task that requires highly experienced consul-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 458–467, Prague, June 2007.

tants to define such categories, which are often described as the viewpoint for doing the analysis, and their corresponding expressions through trial and error.

In this paper, we propose a method to identify important segments of textual data for analysis from full transcripts of conversations. Compared to the written summary of a conversation, a transcription of an entire conversation tends to be quite lengthy and contains various forms of redundancy. Many of the terms appearing in the conversation are not relevant for specific analysis. For example, the terms for greeting such as “Hello” and “Welcome to (Company A)” are unlikely to be associated with specific business results such as purchased-or-not and satisfied-or-not, especially because the conversation is transcribed without preserving the nonverbal moods such as tone of voice, emotion etc. Thus it is crucial to identify key segments and notable expressions within conversations for analysis to acquire valuable insights.

We exploit the fact that business conversations follow set patterns such as an opening followed by a request and the confirmation of details followed by a closing, etc. By taking advantage of this feature of business conversations, we have developed a method to identify key segments and the notable expressions within conversations that tend to discriminate between the business results. Such key segments, the trigger segments, and the notable expressions associated with certain business results lead us to easily understand appropriate viewpoints for analysis.

Application of our method for analyzing nearly one thousand conversations from a rental car reservation office enabled us to acquire novel insights for improving agent productivity and resulted in an actual increase in revenues.

Organization of the Paper: We start by describing the properties of the conversation data used in this paper. Section 3 describes the method for identifying useful viewpoints and expressions that meet the specified purpose. Section 4 provides the results using conversational data. After the discussion in Section 5, we conclude the paper in Section 6.

2 Business-Oriented Conversation Data

We consider business-oriented conversation data collected at contact centers handling inbound telephone sales and reservations. Such business oriented conversations have the following properties.

- Each conversation is a one-to-one interaction between a customer and an agent.
- For many contact center processes the conversation flow is well defined in advance.
- There are a fixed number of outcomes and each conversation has one of these outcomes.

For example, in car rentals, the following conversation flow is pre-defined for the agent. In practice most calls to a car rental center follow this call flow.

- Opening - contains greeting, brand name, name of agent
- Pick-up and return details - agent asks location, dates and times of pick up and return, etc.
- Offering car and rate - agent offers a car specifying rate and mentions applicable special offers.
- Personal details - agent asks for customer’s information such as name, address, etc.
- Confirm specifications - agent recaps reservation information such as name, location, etc.
- Mandatory enquiries - agent verifies clean driving record, valid license, etc.
- Closing - agent gives confirmation number and thanks the customer for calling.

In these conversations the participants speak in turns and the segments can be clearly identified. Figure 1 shows part of a transcribed call.

Each call has a specific outcome. For example, each car rental transaction has one of two call types, reservation or unbooked, as an outcome.

Because the call process is pre-defined, the conversations look similar in spite of having different results. In such a situation, finding the differences in the conversations that have effects on the outcomes is very important, but it is very expensive and difficult to find such unknown differences by human analysis. We show that it is possible to define proper viewpoints and corresponding expressions leading to insights on how to change the outcomes of the calls.

```
AGENT: Welcome to CarCompanyA. My name is Albert.  How may I
help you?
.........
AGENT: Allright may i know the location you want to pick the
car from.
CUSTOMER: Aah ok I need it from SFO.
AGENT: For what date and time.
.........
AGENT: Wonderful so let me see ok mam so we have a 12 or 15
passenger van avilable on this location on those dates and
for that your estimated total for those three dates just
300.58$ this is with Taxes with surcharges and with free
unlimited free milleage.
.........
AGENT : alright mam let me recap the dates you want to pick
it up from SFO on 3rd August and drop it off on august 6th in
LA alright
CUSTOMER : and one more questions Is it just in states or
could you travel out of states
.........
AGENT : The confirmation number for your booking is 221 384.
CUSTOMER : ok ok Thank you
Agent :  Thank you for calling CarCompanyA and you have a
great day good bye
```

Figure 1: Transcript of a car rental dialog (partial)

3 Trigger Segment Detection and Effective Expression Extraction

In this section, we describe a method for automatically identifying valuable segments and concepts from the data for the user-specified difference analysis. First, we present a model to represent the conversational data. After that we introduce a method to detect the segments where the useful concepts for the analysis appear. Finally, we select useful expressions in each detected trigger segment.

3.1 Data Model

Each conversational data record in the collection D is defined as $\boldsymbol{d}_i$. Each $\boldsymbol{d}_i$ can be seen as a sequence of conversational *turns* in the conversational data, and then $\boldsymbol{d}_i$ can be divided as

$$\boldsymbol{d}_i = \boldsymbol{d}_i^1 + \boldsymbol{d}_i^2 + \cdots + \boldsymbol{d}_i^{M_i}, \qquad (1)$$

where $\boldsymbol{d}_i^k$ is the k-th turn in $\boldsymbol{d}_i$ and M_i is the total number of turns in $\boldsymbol{d}_i$. The $+$ operator in the above equation can be seen as an equivalent of the string concatenation operator. We define $\boldsymbol{d}_i^{\sim j}$ as the portion of $\boldsymbol{d}_i$ from the beginning to turn j. Using the same notation, $\boldsymbol{d}_i^{\sim j} = \boldsymbol{d}_i^1 + \boldsymbol{d}_i^2 + \cdots + \boldsymbol{d}_i^j$. The collection of $\boldsymbol{d}_i^{\sim m_k}$ constitutes the *Chronologically Cumulative Data* up to turn m_k (D_k). D_k is represented as

$$D_k = (\boldsymbol{d}_1^{\sim m_k}, \boldsymbol{d}_2^{\sim m_k}, \ldots, \boldsymbol{d}_n^{\sim m_k}). \qquad (2)$$

Figure 2 shows an image of the data model. We set some m_k and prepare the chronologically cumulative data set as shown in Figure 3. We represent binary mutually exclusive business outcomes such as success and failure resulting from the conversations as "A" and "not A".

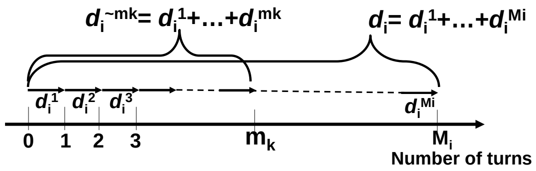


Figure 2: Conversation data model

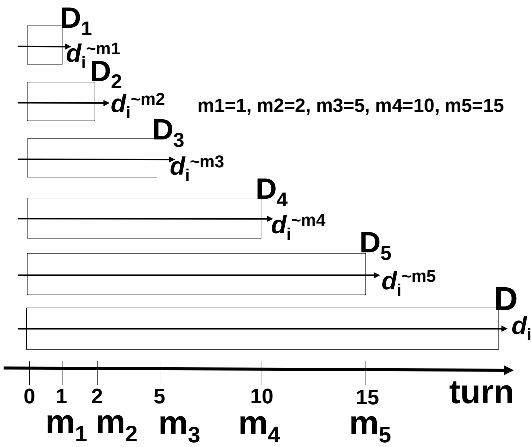


Figure 3: Chronologically cumulative conversational data

3.2 Trigger Segment Detection

Trigger segments can be viewed as portions of the data which have important features which distinguish data of class "A" from data of class "not A".

To detect such segments, we divide each chronologically cumulative data set D_k into two data sets, training data $D_k^{training}$ and test data D_k^{test}. Starting from D_1, for each D_k we trained a classifier using $D_k^{training}$ and evaluated it on D_k^{test}. Using accuracy, the fraction of correctly classified documents, as a metric of performance (Yang and Liu, 1999), we denote the evaluation result of the categorization as $acc(categorizer(D_k))$ for each D_k and plot it along with its turn. Figure 4 shows the effect of gradually increasing the training data for the classification. The distribution of expressions in a business-oriented conversation will change almost synchronously because the call flow is predefined. Therefore $acc(categorizer(D_k))$ will increase if features that contribute to the categorization appear in D_k. In contrast, $acc(categorizer(D_k))$ will decrease if no features that contribute to the categorization are in D_k. Therefore, from the transitions of $acc(categorizer(D_k))$, we can identify the segments with increases as triggers where the features that have an effect on the outcome appear. We denote a trigger segment as $seg(start\, position, end\, position)$. Because the total numbers of turns can be different, we do not detect the last section as a trigger. In Figure 4, $seg(m_1, m_2)$ and $seg(m_4, m_5)$ are triggers. It is important to note that using the cumulative data is key to the detection of trigger segments. Using non-cumulative segment data would give us the categorization accuracy for the features within that segment but would not tell us whether the features of this segment are improving the accuracy or decreasing it. It is this gradient information between segments that is key to identifying trigger segments.

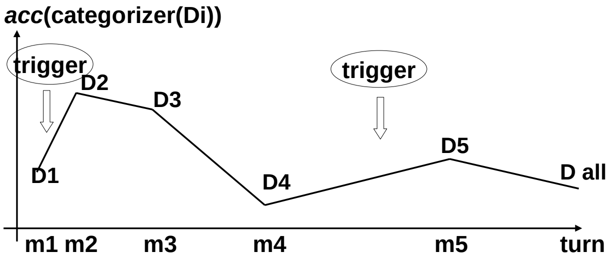


Figure 4: Plot of $acc(categorizer(D_k))$

Many approaches have been proposed for document classification (Yang and Liu, 1999). In this research, however, we are not interested in the classification accuracy itself but in the increase and decrease of the accuracy within particular segments. For example, the greeting, or the particular method of payment may not affect the outcome, but the mention of a specific feature of the product may have an effect on the outcome. Therefore in our research we are interested in identifying the particular portion of the call where this product feature is mentioned, along with its mention, which has an effect on the outcome of the call. In our experiments we used the SVM (Support Vector Machine) classifier (Joachims, 1998), but almost any classifier should work because our approach does not depend on the classification method.

3.3 Effective Expression Extraction

In this section, we describe our method to extract effective expressions from the detected trigger segments.

The effective expressions in D_k are those which are representative in the selected documents and appear for the first time in the trigger segments $seg(m_i, m_j)$. Numerous methods to select features exist (Hisamitsu and Niwa, 2002) (Yang and Pedersen, 1997). We use the χ^2 statistic for each expression in D_k as a representative metric. For the two-by-two contingency table of a expression w and a class "A" shown in Table 1, the χ^2 statistic is calculated as

Table 1: Contingency table for calculating the χ^2 statistic

	# of documents including w	# of documents not including w
A	n_{11}	n_{12}
not-A	n_{21}	n_{22}

$$\chi^2 = \frac{N(n_{11}n_{22} - n_{12}n_{21})^2}{(n_{11}+n_{12})(n_{11}+n_{21})(n_{12}+n_{22})(n_{21}+n_{22})} \quad (3)$$

where N is the number of documents. This statistic can be compared to the χ^2 distribution with one degree of freedom to judge representativeness.

We also want to extract the expressions that have not had an effect on the outcome before D_k. To detect the new expressions in D_k, we define the metric

$$new(w) = \frac{w(D_k)}{\max(w(D_{k-1}), 1)} \Big/ \frac{m_k}{m_{k-1}} \times \mathrm{sign}(w(D_k^A) - w(D_k^{notA})), \quad (4)$$

where $w(D_k)$ is the frequency of expression w in the chronologically cumulative data D_k, $\max(a, b)$ selects the larger value in the arguments, m_k is the number of turns in D_k, $w(D_k^A)$ is the frequency of w in D_k with the outcome of the corresponding data being "A", and $\mathrm{sign}(\cdot)$ is the signum function. When w in class "A" appears in D_k much more frequently than D_{k-1} compared with the ratio of their turns, this metric will be more than 1. We detect significant expressions by considering the combined score $\chi^2(w) \cdot new(w)$. Using this combined score, we can filter out the representative expressions that have already appeared before D_k and distinguish significant expressions that first appear in D_k for each class "A" and "not A".

3.4 Appropriate Viewpoint Selection

In a text mining system, to get an association that leads to a useful insight, we have to define appropriate viewpoints. Viewpoints refer to objects in relation to other objects. In analysis using a conventional text mining system (Nasukawa and Nagano, 2001), the viewpoints are selected based on expressions in user dictionaries prepared by domain experts. We have identified important segments of the conversations by seeing changes in the accuracy of a categorizer designed to segregate different business outcomes. We have also been able to extract effective expressions from these *trigger* segments to define various viewpoints. Hence, viewpoint selection is now based on the trigger segments and effective expressions identified automatically based on specified business outcomes. In the next section we apply our technique to a real life dataset and show that we can successfully select useful viewpoints.

4 Experiments and Results

4.1 Experiment Data and System

We collected 914 recorded calls from the car rental help desk and manually transcribed them. Figure 1 shows part of a call that has been transcribed.

There are three types of calls:

1. **Reservation Calls:** Calls which got *converted*. Here, "converted" means the customer made a reservation for a car. Reserved cars can get *picked-up* or *not picked-up*, so some reserved cars do not eventually get picked-up by customers (no shows and cancellations).

2. **Unbooked Calls:** Calls which did not get converted.

3. **Service Calls:** Customers changing or enquiring about a previous booking.

The distribution of the calls is given in Table 2.

Table 2: Distribution of calls

Unbooked Calls	461
Reservation Calls (Picked-Up)	72
Reservation Calls (Not Picked-Up)	65
Service Calls	326
Total Calls	914

The reservation calls are most important in this context, so we focus on those 137 calls. In the reservation calls, there are two types of outcomes, car picked-up and car not picked-up. All reservation calls look similar in spite of having different outcomes (in terms of pick up). The reservation happens during the call but the pick up happens at a later date. If we can find differences in the conversation that affect the outcome, it is expected that we could improve the agent productivity. Reservation calls follow the pre-defined reservation call flow that we mentioned in Section 2 and it is very difficult to find differences between them manually. In this experiment, by using the proposed method, we try to extract trigger segments and expressions to find viewpoints that affect the outcome of the reservation calls.

For the analysis, we constructed a text mining system for the difference analysis "picked-up" vs. "not picked-up". The experimental system consists of two parts, an information extraction part and a text mining part. In the information extraction part we define dictionaries and templates to identify useful expressions. In the text mining part we define appropriate viewpoints based on the identified expressions to get useful associations leading to useful insights.

4.2 Results of Trigger Segment Detection and Effective Expression Extraction

Based on the pre-defined conversation flow described in Section 2, we set m_1=1, m_2=2, m_3=5, m_4=10, m_5=15, and m_6=20 and prepared $D_1, \ldots, D_6$ and D. The features of $\boldsymbol{d}_i$ consist of nouns, compound nouns, specified noun phrases (e.g. adjective+noun), and verbs. For each D_k we calculated $acc(categorizer(D_k))$ for the classes "picked-up" and "not picked-up." In this process, we use a SVM-based document categorizer (Joachims, 2002). Of the 137 calls, we used 100 calls for training the categorizer and 37 calls for trigger segment detection. Figure 5 shows the results of $acc(categorizer(D_k))$ for picked-up. The accuracy of classification using the data of entire conversations ($acc(categorizer(D))$ is 67.6% but we are trying to detect important segments by considering not the accuracy values themselves but the gradients between segments. From these results, $seg(1, 2)$ and $seg(10, 15)$ are detected as trigger segments. We now know that these segments are highly correlated to the outcome of the call.

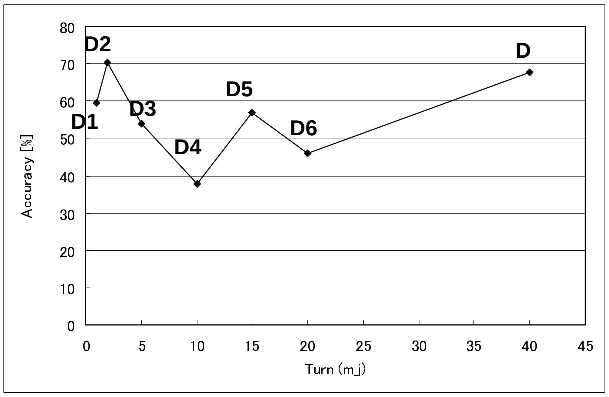


Figure 5: Result of $acc(categorizer(D_k))$

Table 3: Selected expressions in trigger segments

Trigger	Selected expressions	
	pick up	not picked up
$seg(1, 2)$	make, return, tomorrow, day, airport, look, assist, reservation, tonight	rate, check, see want, week
$seg(10, 15)$	number, corporate program, contract, card, have, tax surcharge, just NUMERIC dollars, discount, customer club, good rate, economy	go, impala

For each detected trigger segment, we extract effective expressions in each class using the metric described in Section 3.3. Table 3 shows some expressions with high values for the metric for each trigger. In this table, ”just NUMERIC dollars” is a canonical expression and an expression such as ”just 160 dollars” is mapped to this canonical expression in the information extraction process. From this result, in $seg(1, 2)$, "make", "reservation" are correlated with "pick up" and "rate" and "check" are correlated with "not-picked up". By looking at some documents containing these expressions, we found customer intention phrases such as "would like to make a reservation", "want to check a rate", etc. Therefore, it can be induced that the way a customer starts the call may have an impact on the outcome. From expressions in $seg(10, 15)$, it can be said that discount-related phrases and mentions of the good rates by the agent can have an effect on the outcome.

We can directly apply the conventional methods for representative feature selection to D. The following expressions were selected as the top 20 expressions from whole conversational data by using the χ^2 metric defined in (3).

corporate program, contract, counter, September,
mile, rate, economy, last name,
valid driving license,BRAND NAME, driving,
telephone, midsize, tonight, use, credit, moment,
airline, afternoon

From these results, we see that looking at the call as a whole does not point us to the fact that discount-related phrases, or the first customers-utterance, affect the outcome. Detecting trigger segments and extracting important expressions from each trigger segment are key to identifying subtle differences between very similar looking calls that have entirely opposite outcomes.

4.3 Results of Text Mining Analysis using Selected Viewpoints and Expressions

From the detected segments and expressions we determined that the customer's first utterance along with discount phrases and value selling phrases affected the call outcomes. Under these hypotheses, we prepared the following semantic categories.

- Customer intention at start of call: From the customer's first utterance, we extract the following intentions based on the patterns.
  - strong start: *would like to make a booking, need to pick up a car, . . .*
  - weak start: *would like to check the rates, want to know the rate for vans, . . .*

  Under our hypotheses, the customer with a strong start has the intention of booking a car and we classify such a customer as a **booking_customer**. The customer with a weak start usually just wants to know the rates and is classified as a **rates_customer**.
- discount-related phrases: *discount, corporate program, motor club, buying club . . .* are registered into the domain dictionary as discount-related phrases.
- value selling phrases: we extract phrases mentioning good rates and good vehicles by matching patterns related to such utterances.
  - mentions of good rates: *good rate, wonderful price, save money, just need to pay this low amount, . . .*
  - mentions of good vehicles: *good car, fantastic car, latest model, . . .*

Using these three categories, we tried to find insights to improve agent productivity.

Table 4 shows the result of two-dimensional association analysis for 137 reservation calls. This table shows the association between customer types based on customer intention at the start of a call and pick up information. From these results, 67% (47 out of 70) of the booking_customers picked up the reserved car and only 35% (13 out of 37) of the rates_customers picked it up. This supports our hypothesis and means that pick up is predictable from the customer's first or second utterance.

Table 4: Association between customer types and pick up information

Customer types extracted from texts based on customer intent at start of call	Pick up information	
	pick up	not-picked up
booking_customer (w/ strong start) (70)	47	23
rates_customer (w/ weak start) (37)	13	24

It was found that cars booked by rates_customers tend to be "not picked up," so if we can find any actions by agents that convert such customers into "pick up," then the revenue will improve. In the booking_customer case, to keep the "pick up" high, we need to determine specific agent actions that concretize the customer's intent.

Table 5 shows how mentioning discount-related phrases affects the pick up ratios for rates_customers and booking_customers. From this table, it can be seen that mentioning discount phrases affects the final status of both types of customers. In the rates_customer case, the probability that the booked car will be picked up, $P(\text{pick-up})$ is improved to 0.476 by mentioning discount phrases. This means customers are attracted by offering discounts and this changes their intention from "just checking rate" to "make a reservation here". We found similar trends for the association between mention of value selling phrases and pick up information.

Table 5: Association between mention of discount phrases and pick up information

$Rates\_customer$	Pick up information	
Mention of discount phrases by agents	pick up	not-picked up
yes (21)	10	11
no (16)	3	13

$Booking\_customer$	Pick up information	
Mention of discount phrases by agents	pick up	not picked up
yes (40)	30	10
no (30)	17	13

4.4 Improving Agent Productivity

From the results of the text mining analysis experiment, we derived the following actionable insights:

- There are two types of customers in reservation calls.
  - **Booking_customer** (with strong start) tends to pick up the reserved car.
  - **Rates_customer** (with weak start) tends not to pick up the reserved car.
- In the **rates_customer** case, "pick up" is improved by mentioning discount phrases.

By implementing the actionable insights derived from the analysis in an actual car rental process, we verified improvements in pick up. We divided the 83 agents in the car rental reservation center into two groups. One of them, consisting of 22 agents, was trained based on the insights from the text mining analysis. The remaining 61 agents were not told about these findings. By comparing these two groups over a period of one month we hoped to see how the actionable insights contributed to improving agent performance. As the evaluation metric, we used the pick up ratio - that is the ratio of the number of “pick-ups” to the number of reservations.

Following the training the pick up ratio of the trained agents increased by 4.75%. The average pick up ratio for the remaining agents increased by 2.08%. Before training the ratios of both groups were comparable. The seasonal trends in this industry mean that depending on the month the bookings and pickups may go up or down. We believe this is why the average pick up ratio for the remaining agents also increased. Considering this, it can be estimated that by implementing the actionable insights the pick up ratio for the pilot group was improved by about 2.67%. We confirmed that this difference is meaningful because the p-value of the t-test statistic is 0.0675 and this probability is close to the standard t-test (α=0.05). Seeing this, the contact center trained all of its agents based on the insights from the text mining analysis.

5 Discussion

There has been a lot of work on specific tools for analyzing the conversational data collected at contact centers. These include call type classification for the purpose of categorizing calls (Tang et al., 2003) (Zweig et al., 2006), call routing (Kuo and Lee, 2003) (Haffner et al., 2003), obtaining call log summaries (Douglas et al., 2005), agent assisting and monitoring (Mishne et al., 2005), and building of domain models (Roy and Subramaniam, 2006). Filtering problematic dialogs automatically from an automatic speech recognizer has also been studied (Hastie et al., 2002) (Walker et al., 2002). In contrast to these technologies, in this paper we consider the task of trying to find insights from a collection of complete conversations. In (Nasukawa and Nagano, 2001), such an analysis was attempted for agent-entered call summaries of customer contacts by extracting phrases based on domain-expert-specified viewpoints. In our work we have shown that even for conversational data, which is more complex, we could identify proper viewpoints and prepare expressions for each viewpoint. Call summaries by agents tend to mask the customers’ intention at the start of the call. We get more valuable insights from the text mining analysis of conversational data. For such an analysis of conversational data, our proposed method has an important role. With our method, we find the important segments in the data for doing analyses. Also our analyses are closely linked to the desired outcomes.

In trigger detection, we created a chronologically cumulative data set based on turns. We can also use the segment information such as the “opening” and “enquiries” described in Section 2. We prepared data with segment information manually assigned, made the chronologically cumulative data and applied our trigger detection method. Figure 6 shows the results of $acc(categorizer(D_k))$. The trend in Figure 6 is similar to that in Figure 5. From this result, it is observed that ”opening” and ”offering” segments are trigger segments. Usually, segmentation is not done in advance and to assign such information automatically we need data with labeled segmentation information. The results show that even in the absence of labeled data our trigger detection method identifies the trigger segments. In the experiments in Section 4, we set turns for each chronologically cumulative data by taking into account the pre-defined call flow.

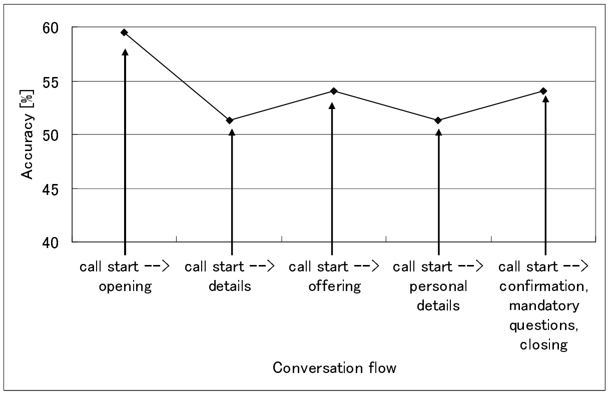


Figure 6: Result of $acc(categorizer(D_k))$ using segment information

In Figure 5 we observe that the accuracy of the categorizer is decreasing even when using increasing parts of the call. Even the accuracy using the complete call is less than using only the first turn. This indicates that the first turn is very informative, but it also indicates that the *features* are not being used judiciously. In a conventional classification task, the number of features are sometimes restricted when constructing a categorizer. It is known that selecting only significant features improves the classification accuracy (Yang and Pedersen, 1997). We used *Information Gain* for selecting features from the document collection. This method selects the most discriminative features between two classes. As expected the classification accuracy improved significantly as we reduced the total number of features from over 2,000 to the range of 100 to 300. Figure 7 shows the changes in accuracy. In the proposed method, we detect trigger segments using the increases and decreases of the classification accuracy. By selecting features, the noisy features are not added in the segments. Decreasing portions, therefore are not observed. In this situation, as a trigger segment, we can detect the portion where the gradient of the accuracy curve increases. Also using feature selection, we find that the classification accuracy is highest when using the entire document, which is expected. However, we notice that the trigger segments obtained with and without feature selection are almost the same.

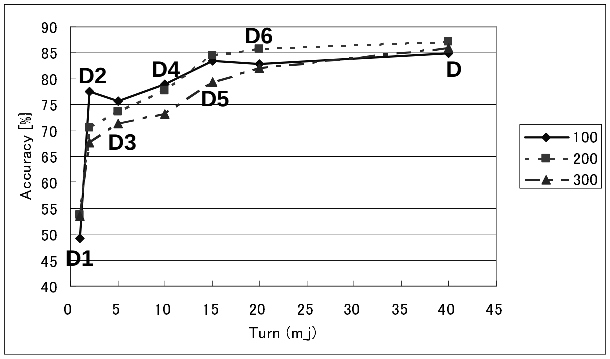


Figure 7: Result of $acc(categorizer(D_k))$ with top 100 to 300 features selected using information gain

In the experiment, we use manually transcribed data. As future work we would like to use the noisy output of an automatic speech recognition system to obtain viewpoints and expressions.

6 Conclusion

In this paper, we have proposed methods for identifying appropriate segments and expressions automatically from the data for user specified difference analysis. We detected the trigger segments using the property that a business-oriented conversation follows a pre-defined flow. After that, we identified the appropriate expressions from each trigger segment. It was found that in a long business-priented conversation there are important segments affecting the outcomes that can not been easily detected by just looking through the conversation, but such segments can be detected by monitoring the changes of the categorization accuracy. For the trigger segment detection, we do not use semantic segment information but only the positional segment information based on the conversational turns. Because our method does not rely on the semantic information in the data, therefore our method can be seen as robust. Through experiments with real conversational data, using identified segments and expressions we were able to define appropriate viewpoints and concepts leading to insights for improving the car rental business process.

Acknowledgment

The authors would like to thank Sreeram Balakrishnan, Raghuram Krishnapuram, Hideo Watanabe, and Koichi Takeda at IBM Research for their support. The authors also appreciate the efforts of Jatin Joy Giri at IBM India in providing domain knowledge about the car rental process and thank him for help in constructing the dictionaries.

References

S. Douglas, D. Agarwal, T. Alonso, R. M. Bell, M. Gilbert, D. F. Swayne, and C. Volinsky. 2005. Mining customer care dialogs for “daily news”. *IEEE Transaction on Speech and Audio Processing*, 13(5):652–660.

P. Haffner, G. Tur, and J. H. Wright. 2003. Optimizing svms for complex call classification. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 632–635.

H. W. Hastie, R. Prasad, and M. A. Walker. 2002. What’s the trouble: Automatically identifying problematic dialogues in darpa communicator dialogue systems. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 384–391.

T. Hisamitsu and Y. Niwa. 2002. A measure of term representativeness based on the number of co-occurring sailent words. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 1–7.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142.

H.-K J. Kuo and C.-H. Lee. 2003. Discriminative training of natural language call routers. *IEEE Transaction on Speech and Audio Processing*, 11(1):24–35.

G. Mishne, D. Carmel, R. Hoory, A. Roytman, and A. Soffer. 2005. Automatic analysis of call-center conversations. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 453–459.

T. Nasukawa and T. Nagano. 2001. Text analysis and knowledge mining system. *IBM Systems Journal*, pages 967–984.

S. Roy and L. V. Subramaniam. 2006. Automatic generation of domain models for call centers from noisy transcriptions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (COLING/ACL)*, pages 737–744.

M. Tang, B. Pellom, and K. Hacioglu. 2003. Call-type classification and unsupervised training for the call center domain. In *Proceesings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 204–208.

M. A. Walker, I. Langkilde-Geary, H. W. Hastie, J. Wright, and A. Gorin. 2002. Automatically training a problematic dialogue predictor for a spoken dialogue system. *Journal of Artificial Intelligence Research*, 16:393–319.

Y. Yang and X. Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49.

Y. Yang and J. O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 412–420.

G. Zweig, O. Shiohan, G. Saon, B. Ramabhadran, D. Povey, L. Mangu, and B. Kingsbury. 2006. Automatic analysis of call-center conversations. In *Proceedings of IEEE Internatinal Conference of Acoustics, Speech and Signal Processing (ICASSP)*, pages 589–592.

Smoothed Bloom filter language models: Tera-Scale LMs on the Cheap

David Talbot and Miles Osborne
School of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW, UK
d.r.talbot@sms.ed.ac.uk, miles@inf.ed.ac.uk

Abstract

A Bloom filter (BF) is a randomised data structure for set membership queries. Its space requirements fall significantly below lossless information-theoretic lower bounds but it produces false positives with some quantifiable probability. Here we present a general framework for deriving smoothed language model probabilities from BFs.

We investigate how a BF containing n-gram statistics can be used as a *direct replacement* for a conventional n-gram model. Recent work has demonstrated that corpus statistics can be stored efficiently within a BF, here we consider how smoothed language model probabilities can be derived efficiently from this randomised representation. Our proposal takes advantage of the one-sided error guarantees of the BF and simple inequalities that hold between related n-gram statistics in order to further reduce the BF storage requirements and the error rate of the derived probabilities. We use these models as replacements for a conventional language model in machine translation experiments.

1 Introduction

Language modelling (LM) is a crucial component in statistical machine translation (SMT). Standard n-gram language models assign probabilities to translation hypotheses in the target language, typically as smoothed trigram models (Chiang, 2005). Although it is well-known that higher-order language models and models trained on additional monolingual corpora can significantly improve translation performance, deploying such language models is not trivial. Increasing the order of an n-gram model can result in an exponential increase in the number of parameters; for the English Gigaword corpus, for instance, there are 300 million distinct trigrams and over 1.2 billion distinct five-grams. Since a language model is potentially queried millions of times per sentence, it should ideally reside locally in memory to avoid time-consuming remote or disk-based lookups.

Against this background, we consider a radically different approach to language modelling. Instead of explicitly storing all distinct n-grams from our corpus, we create an implicit randomised representation of these statistics. This allows us to drastically reduce the space requirements of our models. In this paper, we build on recent work (Talbot and Osborne, 2007) that demonstrated how the *Bloom filter* (Bloom (1970); BF), a space-efficient randomised data structure for representing sets, could be used to store corpus statistics efficiently. Here, we propose a framework for deriving smoothed n-gram models from such structures and show via machine translation experiments that these *smoothed Bloom filter language models* may be used as direct replacements for standard n-gram models in SMT.

The space requirements of a Bloom filter are quite spectacular, falling significantly below information-theoretic error-free lower bounds. This efficiency, however, comes at the price of *false positives*: the filter may erroneously report that an item not in the set is a member. False negatives, on the other hand, will

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 468–476, Prague, June 2007. ©2007 Association for Computational Linguistics

never occur: the error is said to be *one-sided*. Our framework makes use of the *log-frequency Bloom filter* presented in (Talbot and Osborne, 2007), and described briefly below, to compute smoothed conditional n-gram probabilities on the fly. It takes advantage of the one-sided error guarantees of the Bloom filter and certain inequalities that hold between related n-gram statistics drawn from the same corpus to reduce both the error rate and the computation required in deriving these probabilities.

2 The Bloom filter

In this section, we give a brief overview of the Bloom filter (BF); refer to Broder and Mitzenmacher (2005) for a more in detailed presentation. A BF represents a set $\mathcal{S} = \{x_1, x_2, ..., x_n\}$ with n elements drawn from a universe $\mathcal{U}$ of size N. The structure is attractive when $N \gg n$. The only significant storage used by a BF consists of a bit array of size m. This is initially set to hold zeroes. To train the filter we hash each item in the set k times using distinct hash functions $h_1, h_2, ..., h_k$. Each function is assumed to be independent from each other and to map items in the universe to the range 1 to m uniformly at random. The k bits indexed by the hash values for each item are set to 1; the item is then discarded. Once a bit has been set to 1 it remains set for the lifetime of the filter. Distinct items may not be hashed to k distinct locations in the filter; we ignore collisons. Bits in the filter can, therefore, be *shared* by distinct items allowing significant space savings but introducing a non-zero probability of false positives at test time. There is no way of directly retrieving or ennumerating the items stored in a BF.

At test time we wish to discover whether a given item was a member of the original set. The filter is queried by hashing the test item using the same k hash functions. If all bits referenced by the k hash values are 1 then we *assume* that the item was a member; if any of them are 0 then we *know* it was not. True members are always correctly identified, but a false positive will occur if all k corresponding bits were set by other items during training and the item was not a member of the training set.

The probability of a false postive, f, is clearly the probability that none of k randomly selected bits in the filter are still 0 after training. Letting p be the proportion of bits that are still zero after these n elements have been inserted, this gives,

$$f = (1 - p)^k.$$

As n items have been entered in the filter by hashing each k times, the probability that a bit is still zero is,

$$p^{'} = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}}$$

which is the expected value of p. Hence the false positive rate can be approximated as,

$$f = (1 - p)^k \approx (1 - p^{'})^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k.$$

By taking the derivative we find that the number of functions k^* that minimizes f is,

$$k^* = \ln 2 \cdot \frac{m}{n},$$

which leads to the intuitive result that exactly half the bits in the filter will be set to 1 when the optimal number of hash functions is chosen.

The fundmental difference between a Bloom filter's space requirements and that of any lossless representation of a set is that the former does not depend on the size of the (exponential) universe N from which the set is drawn. A lossless representation scheme (for example, a hash map, trie etc.) must depend on N since it assigns a distinct representation to each possible set drawn from the universe.

3 Language modelling with Bloom filters

Recent work (Talbot and Osborne, 2007) presented a scheme for associating static frequency information with a set of n-grams in a BF efficiently.[1]

3.1 Log-frequency Bloom filter

The efficiency of the scheme for storing n-gram statistics within a BF presented in Talbot and Osborne (2007) relies on the Zipf-like distribution of n-gram frequencies: most events occur an extremely small number of times, while a small number are very frequent. We assume that raw counts are quantised and employ a logarithmic codebook that maps counts, $c(x)$, to quantised counts, $qc(x)$, as follows,

$$qc(x) = 1 + \lfloor \log_b c(x) \rfloor. \quad (1)$$

[1] Note that as described the Bloom filter is *not* an associative data structure and provides only a Boolean function characterising the set that has been stored in it.

Algorithm 1 Training frequency BF

```
Input: $\mathcal{S}_{train}$, $\{h_1, ...h_k\}$ and $\mathcal{BF} = \emptyset$
Output: $\mathcal{BF}$
for all $x \in \mathcal{S}_{train}$ do
    $c(x) \leftarrow$ frequency of $n$-gram $x$ in $\mathcal{S}_{train}$
    $qc(x) \leftarrow$ quantisation of $c(x)$ (Eq. 1)
    for $j = 1$ to $qc(x)$ do
        for $i = 1$ to $k$ do
            $h_i(x) \leftarrow$ hash of event $\{x, j\}$ under $h_i$
            $\mathcal{BF}[h_i(x)] \leftarrow 1$
        end for
    end for
end for
return $\mathcal{BF}$
```

The precision of this codebook decays exponentially with the raw counts and the scale is determined by the base of the logarithm b; we examine the effect of this parameter on our language models in experiments below.

Given the quantised count $qc(x)$ for an n-gram x, the filter is trained by entering composite events consisting of the n-gram appended by an integer counter j that is incremented from 1 to $qc(x)$ into the filter. To retrieve an n-gram's frequency, the n-gram is first appended with a counter set to 1 and hashed under the k functions; if this tests positive, the counter is incremented and the process repeated. The procedure terminates as soon as any of the k hash functions hits a 0 and the previous value of the counter is reported. The one-sided error of the BF and the training scheme ensure that the actual quantised count cannot be larger than this value. As the counts are quantised logarithmically, the counter is usually incremented only a small number of times.

We can then approximate the original frequency of the n-gram by taking its expected value given the quantised count retrieved,

$$E[c(x)|qc(x) = j] = \frac{b^{j-1} + b^j - 1}{2}. \quad (2)$$

These training and testing routines are repeated here as Algorithms 1 and 2 respectively.

As noted in Talbot and Osborne (2007), errors for this log-frequency BF scheme are one-sided: frequencies will never be underestimated. The probability of overestimating an item's frequency decays exponentially with the size of the overestimation error d (i.e. as f^d for $d > 0$) since each erroneous increment corresponds to a single false positive and d such independent events must occur together.

Algorithm 2 Test frequency BF

```
Input: $x$, $MAXQCOUNT$, $\{h_1, ...h_k\}$ and $\mathcal{BF}$
Output: Upper bound on $c(x) \in \mathcal{S}_{train}$
for $j = 1$ to $MAXQCOUNT$ do
    for $i = 1$ to $k$ do
        $h_i(x) \leftarrow$ hash of event $\{x, j\}$ under $h_i$
        if $\mathcal{BF}[h_i(x)] = 0$ then
            return $E[c(x)|qc(x) = j - 1]$ (Eq. 2)
        end if
    end for
end for
```

The efficiency of the log-frequency BF scheme can be understood from an entropy encoding perspective under the distribution over frequencies of n-gram types: the most common frequency (the singleton count) is assigned the shortest code (length k) while rarer frequencies (those for more common n-grams) are assigned increasingly longer codes ($k \times qc(x)$).

3.2 Smoothed BF language models

A standard n-gram language model assigns conditional probabilities to target words given a certain context. In practice, most standard n-gram language models employ some form of interpolation whereby probabilities conditioned on the most specific context consisting usually of the $n - 1$ preceding tokens are combined with more robust estimates based on less specific conditioning events. To compute smoothed language model probabilities, we generally require access to the frequencies of n-grams of length 1 to n in our training corpus. Depending on the smoothing scheme, we may also need auxiliary statistics regarding the number of distinct suffixes for each n-gram (e.g., Witten-Bell and Kneser-Ney smoothing) and the number of distinct prefixes or contexts in which they appear (e.g., Kneser-Ney). We can use a single BF to store these statistics but need to distinguish each type of event (e.g., raw counts, suffix counts, etc.). Here we use a distinct set of k hash functions for each such category.

Our motivation for storing the corpus statistics directly rather than precomputed probabilities is twofold: (i) the efficiency of the scheme described above for storing frequency information together with items in a BF relies on the frequencies having a Zipf-like distribution; while this is definitely true for corpus statistics, it may well not hold for probabilities estimated from them; (ii) as will become apparent below, by using the corpus statistics directly, we will be able to make additional savings in terms of both space and error rate by using simple inequalities that hold for related information drawn consistently from the same corpus; it is not clear whether such bounds can be established for probabilities computed from these statistics.

3.2.1 Proxy items

There is a potential risk of redundancy if we represent related statistics using the log-frequency BF scheme presented in Talbot and Osborne (2007). In particular, we do not need to store information explicitly that is necessarily implied by the presence of another item in the training set, if that item can be identified efficiently at query time when needed. We use the term *proxy item* to refer to items whose presence in the filter implies the existence of another item *and* that can be efficiently queried given the implied item. In using a BF to store corpus statistics for language modelling, for example, we can use the event corresponding to an n-gram and the counter set to 1 as a proxy item for a distinct prefix, suffix or context count of 1 for the same n-gram since (ignoring sentence boundaries) it must have been preceded and followed by at least one distinct type, i.e.,

$$qc(w_1, ..., w_n) \geq 1 \in BF \Rightarrow s(w_1, ..., w_n) \geq 1,$$

where $s(\cdot)$ is the number of the distinct types following this n-gram in the training corpus. We show below that such lower bounds allow us to significantly reduce the memory requirements for a BF language model.

3.2.2 Monotonicity of n-gram event space

The error analysis in Section 2 focused on the false positive rate of a BF; if we deploy a BF within an SMT decoder, however, the actual error rate will also depend on the *a priori* membership probability of items presented to it. The error rate Err is,

$$Err = Pr(x \notin S_{train}|Decoder)f.$$

This implies that, unlike a conventional lossless data structure, the model's accuracy depends on other components in system and how it is queried.

Assuming that statistics are entered consistently from the same corpus, we can take advantage of the monotonicity of the n-gram event space to place upper bounds on the frequencies of events to be retrieved from the filter prior to querying it, thereby reducing the *a priori* probability of a negative and consequently the error rate.

Specifically, since the log-frequency BF scheme will never underestimate an item's frequency, we can apply the following inequality recursively and bound the frequency of an n-gram by that of its least frequent subsequence,

$$c(w_1, ..., w_n) \leq \min\{c(w_1, ..., w_{n-1}), c(w_2, ..., w_n)\}.$$

We use this to reduce the error rate of an interpolated BF language model described below.

3.3 Witten-Bell smoothed BF LM

As an example application of our framework, we now describe a scheme for creating and querying a log-frequency BF to estimate n-gram language model probabilities using Witten-Bell smoothing (Bell et al., 1990). Other smoothing schemes, notably Kneser-Ney, could be described within this framework using additional proxy relations for infix and prefix counts.

In Witten-Bell smoothing, an n-gram's probability is discounted by a factor proportional to the number of times that the $n-1$-gram preceding the current word was observed preceding a novel type in the training corpus. It is defined recursively as,

$$P_{wb}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{ml}(w_i|w_{i-n+1}^{i-1}) \\ + (1-\lambda_{w_{i-n+1}^{i-1}}) P_{wb}(w_i|w_{i-n+2}^{i-1})$$

where λ_x is defined via,

$$1 - \lambda_x = \frac{c(x)}{s(x) + c(x)},$$

and $P_{ml}(\cdot)$ is the maximum likelihood estimator calculated from relative frequencies.

The statistics required to compute the Witten-Bell estimator for the conditional probability of an n-gram consist of the counts of all n-grams of length

1 to n as well as the counts of the number of distinct types following all n-grams of length 1 to $n - 1$. In practice we use the $c(w_1, ..., w_i) = 1$ event as a proxy for $s(w_1, ..., w_i) = 1$ and thereby need not store singleton suffix counts in the filter.

Distinct suffix counts of 2 and above are stored by subtracting this proxy count and converting to the log quantisation scheme described above, i.e.,

$$qs(x) = 1 + \lfloor \log_b(s(x) - 1) \rfloor$$

In testing for a suffix count, we first query the item $c(w_1, ..., w_n) = 1$ as a proxy for $s(w_1, ..., w_n) = 1$ and, if found, query the filter for incrementally larger suffix counts, taking the reconstructed suffix count of an n-gram with a non-zero n-gram count to be the expected value, i.e.,

$$E[s(x)|qs(x) = j \cap j > 0] = 1 + \frac{(b^{j-1} + b^j - 1)}{2}$$

Having created a BF containing these events, the algorithm we use to compute the interpolated WB estimate makes use of the inequalities described above to reduce the *a priori* probability of querying for a negative. In particular, we bound the count of each numerator in the maximum likelihood term by the count of the corresponding denominator and the count of distinct suffixes of an n-gram by its respective token frequency.

Unlike more traditional LM formulations that back-off from the highest-order to lower-order models, our algorithm works up from the lowest-order model. Since the conditioning context increases in specificity at each level, each statistic is bound from above by its corresponding value at the previous less specific level. The bounds are applied by passing them as the parameter $MAXQCOUNT$ to the frequency test routine shown as Algorithm 2. We analyze the effect of applying such bounds on the performance of the model within an SMT decoder in the experiments below. Working upwards from the lower-order models also allows us to truncate the computation before the highest level if the denominator in the maximum likelihood term is found with a zero count at any stage (no higher-order terms can be non-zero given this).

4 Experiments

We conducted a range of experiments to explore the error-space trade-off of using a BF-based model as a replacement for a conventional n-gram model within an SMT system and to assess the benefits of specific features of our framework for deriving language model probabilities from a BF.

4.1 Experimental set-up

All of our experiments use publically available resources. Our main experiments use the French-English section of the *Europarl* (EP) corpus for parallel data and language modelling (Koehn, 2003). Decoding is carried-out using the Moses decoder (Koehn and Hoang, 2007). We hold out 1,000 test sentences and 500 development sentences from the parallel text for evaluation purposes. The parameters for the feature functions used in this log-linear decoder are optimised using minimum error rate (MER) training on our development set unless otherwise stated. All evaluation is in terms of the BLEU score on our test set (Papineni et al., 2002).

Our baseline language models were created using the SRILM toolkit (Stolcke, 2002). We built 3, 4 and 5-gram models from the Europarl corpus using interpolated Witten-Bell smoothing (WB); no n-grams are dropped from these models or any of the BF-LMs. The number of distinct n-gram types in these baseline models as well as their sizes on disk and as compressed by *gzip* are given in Table 1; the *gzip* figures are given as an approximate (and optimistic) lower bound on lossless representations of these models.[2]

The BF-LM models used in these experiments were all created from the same corpora following the scheme outlined above for storing n-gram statistics. Proxy relations were used to reduce the number of items that must be stored in the BF; in addition, unless specified otherwise, we take advantage of the bounds described above that hold between related statistics to avoid presenting known negatives to the filter. The base of the logarithm used in quantization is specified on all figures.

The SRILM and BF-based models are both queried via the same interface in the Moses decoder.

[2] Note, in particular, that *gzip* compressed files do *not* support direct random access as required by in language modelling.

n	Types	Mem.	Gzip'd	BLEU
3	5.9M	174Mb	51Mb	28.54
4	14.1M	477Mb	129Mb	28.99
5	24.2M	924Mb	238Mb	29.07

Table 1: WB-smoothed SRILM baseline models.

We assign a small cache to the BF-LM models (between 1 and 2MBs depending on the order of the model) to store recently retrieved statistics and derived probabilities. Translation takes between 2 to 5 times longer using the BF-LMs as compared to the corresponding SRILM models.

4.2 Machine translation experiments

Our first set of experiments examines the relationship between memory allocated to the BF-LM and translation performance for a 3-gram and a 5-gram WB smoothed BF-LM. In these experiments we use the log-linear weights of the baseline model to avoid variation in translation performance due to differences in the solutions found by MER training: this allows us to focus solely on the quality of each BF-LM's approximation of the baseline. These experiments consider various settings of the base for the logarithm used during quantisation (b in Eq. (1)).

We also analyse these results in terms of the relationships between BLEU score and the underlying error rate of the BF-LM and the number of bits assigned per n-gram in the baseline model.

MER optimised BLEU scores on the test set are then given for a range of BF-LMs.

4.3 Mean squared error experiments

Our second set of experiments focuses on the accuracy with which the BF-LM can reproduce the baseline model's distribution. Unfortunately, perplexity or related information-theoretic quantities are not applicable in this case since the BF-LM is not guaranteed to produce a properly normalised distribution. Instead we evaluate the mean squared error (MSE) between the log-probabilites assigned by the baseline model and by BF-LMs to n-grams in the English portion of our development set; we also consider the relation between MSE and the BLEU score from the experiments above.

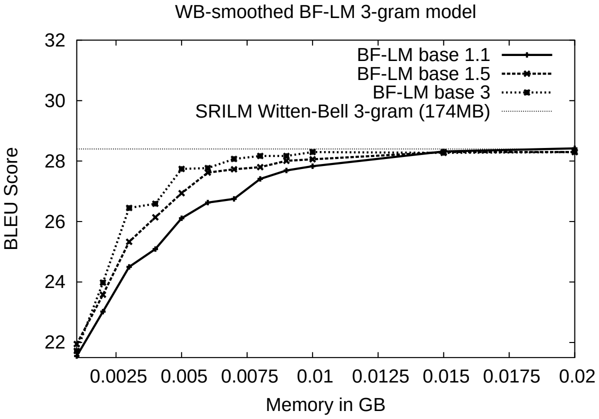


Figure 1: WB-smoothed 3-gram model (Europarl).

4.4 Analysis of BF-LM framework

Our third set of experiments evaluates the impact of the use of upper bounds between related statistics on translation performance. Here the standard model that makes use of these bounds to reduce the *a priori* negative probability is compared to a model that queries the filter in a memoryless fashion.[3]

We then present details of the memory savings obtained by the use of proxy relations for the models used here.

5 Results

5.1 Machine translation experiments

Figures 1 and 2 show the relationship between translation performance as measured by BLEU and the memory assigned to the BF respectively for WB-smoothed 3-gram and 5-gram BF-LMs. There is a clear degradation in translation performance as the memory assigned to the filter is reduced. Models using a higher quantisation base approach their optimal performance faster; this is because these more coarse-grained quantisation schemes store fewer items in the filter and therefore have lower underlying false positive rates for a given amount of memory.

Figure 3 presents these results in terms of the relationship between translation performance and the false positive rate of the underlying BF. We can see that for a given false positive rate, the more coarse-grained quantisation schemes (e.g., base 3) perform

[3] In both cases we apply 'sanity check' bounds to ensure that none of the ratios in the WB formula (Eq. 3) are greater than 1.

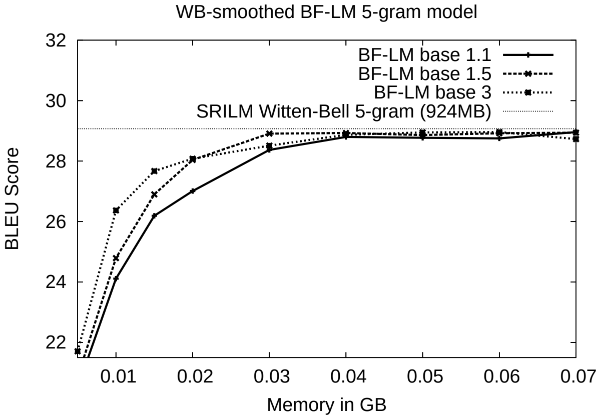


Figure 2: WB-smoothed 5-gram model (Europarl).

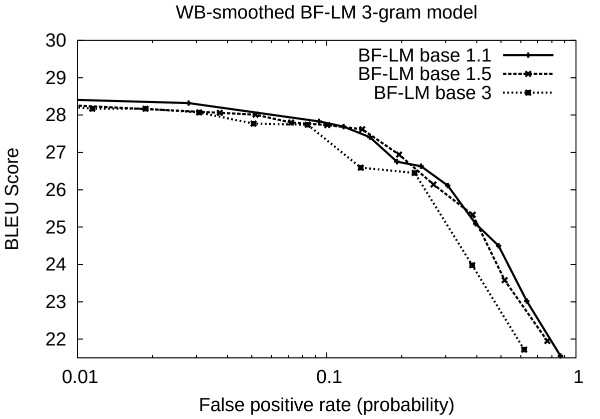


Figure 3: False positive rate vs. BLEU .

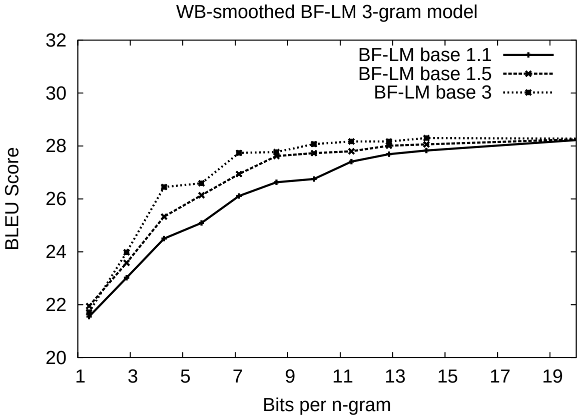


Figure 4: Bits per n-gram vs. BLEU.

n	Memory	Bits / n-gram	base	BLEU
3	10MB	14 bits	1.5	28.33
3	10MB	14 bits	2.0	28.47
4	20MB	12 bits	1.5	28.63
4	20MB	12 bits	2.0	28.63
5	40MB	14 bits	1.5	28.53
5	40MB	14 bits	2.0	28.72
5	50MB	17 bits	1.5	29.31
5	50MB	17 bits	2.0	28.67

Table 2: MERT optimised WB-smoothed BF-LMS.

worse than the more fine-grained schemes.[4]

Figure 4 presents the relationship in terms of the number of bits per n-gram in the baseline model. This suggests that between 10 and 15 bits is sufficient for the BF-LM to approximate the baseline model. This is a reduction of a factor of between 16 and 24 on the plain model and of between 4 and 7 on *gzip* compressed model.

The results of a selection of BF-LM models with decoder weights optimised using MER training are given in Table 2; these show that the models perform consistently close to the baseline models that they approximate.

5.2 Mean squared error experiments

Figure 5 shows the relationship between memory assigned to the BF-LMs and the mean squared error (MSE) of log-probabilities that these models assign to the development set compared to those assigned by the baseline model. This shows clearly that the more fine-grained quantisation scheme (e.g. base 1.1) can reach a lower MSE but also that the more coarse-grained schemes (e.g., base 3) approach their minimum error faster.

Figure 6 shows the relationship between MSE between the BF-LM and the baseline model and BLEU. The MSE appears to be a good predictor of BLEU score across all quantisation schemes. This suggests that it may be a useful tool for optimising BF-LM parameters without the need to run the decoder assuming a target (lossless) LM can be built and queried for a small test set on disk. An MSE of below 0.05 appears necessary to achieve translation performance matching the baseline model here.

5.3 Analysis of BF-LM framework

We refer to (Talbot and Osborne, 2007) for empirical results establishing the performance of the log-frequency BF-LM: overestimation errors occur with

[4] Note that in this case the base 3 scheme will use approximately two-thirds the amount of memory required by the base 1.5 scheme.

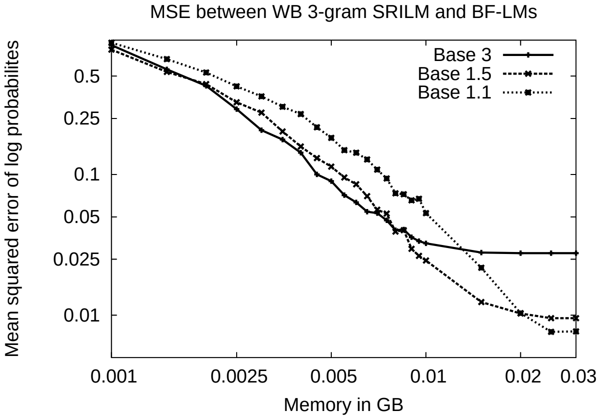


Figure 5: MSE between SRILM and BF-LMs

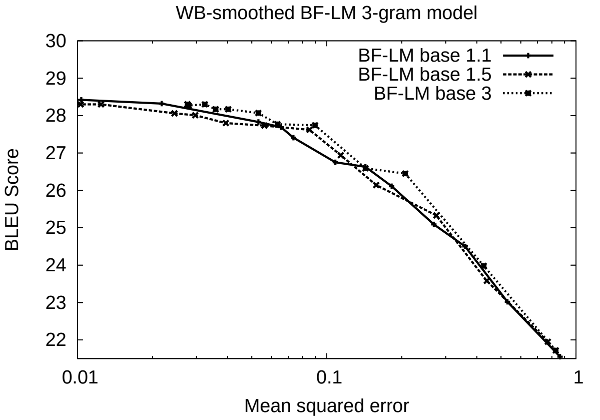


Figure 6: MSE vs. BLEU for WB 3-gram BF-LMs

a probability that decays exponentially in the size of the overestimation error.

Figure 7 shows the effect of applying upper bounds to reduce the *a priori* probability of presenting a negative event to the filter in our interpolation algorithm for computing WB-smoothed probabilities. The application of upper bounds improves translation performance particularly when the amount of memory assigned to the filter is limited. Since both filters have the same underlying false positive rate (they are identical), we can conclude that this improvement in performance is due to a reduction in the number of negatives that are presented to the filter and hence errors.

Table 3 shows the amount of memory saved by the use of proxy items to avoid storing singleton suffix counts for the Witten-Bell smoothing scheme. The savings are given as ratios over the amount of memory needed to store the statistics without proxy items. These models have the same underlying false positive rate (0.05) and quantisation base (2). Similar savings may be anticipated when applying this framework to infix and prefix counts for Kneser-Ney smoothing.

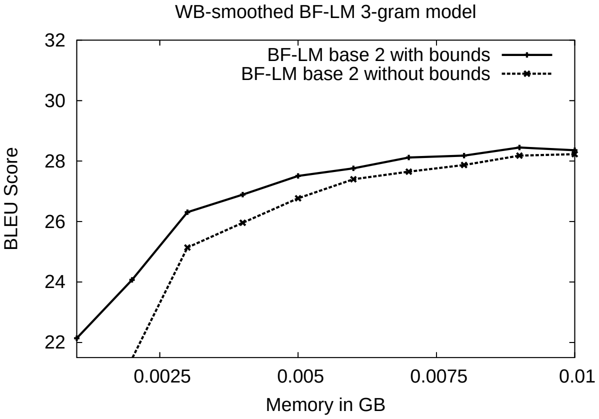


Figure 7: Effect of upper bounds on BLEU

n-gram order	Proxy space saving
3	0.885
4	0.783
5	0.708

Table 3: Space savings via proxy items .

6 Related Work

Previous work aimed at reducing the size of n-gram language models has focused primarily on quantisation schemes (Whitaker and Raj, 2001) and pruning (Stolcke, 1998). The impact of the former seems limited given that storage for the n-gram types themselves will generally be far greater than that needed for the actual probabilities of the model. Pruning on the other hand could be used in conjunction with the framework proposed here. This holds also for compression schemes based on clustering such as (Goodman and Gao, 2000). Our approach, however, avoids the significant computational costs involved in the creation of such models.

Other schemes for dealing with large language models include per-sentence filtering of the model or its distribution over a cluster. The former requires time-consuming adaptation of the model for each sentence in the test set while the latter incurs significant overheads for remote calls during decoding. Our framework could, however, be used to complement either of these approaches.

7 Conclusions and Future Work

We have proposed a framework for computing smoothed language model probabilities efficiently from a randomised representation of corpus statistics provided by a Bloom filter. We have demonstrated that models derived within this framework can be used as direct replacements for equivalent conventional language models with significant reductions in memory requirements. Our empirical analysis has also demonstrated that by taking advantage of the one-sided error guarantees of the BF and simple inequalities that hold between related n-gram statistics we are able to further reduce the BF storage requirements and the effective error rate of the derived probabilities.

We are currently implementing Kneser-Ney smoothing within the proposed framework. We hope the present work will, together with Talbot and Osborne (2007), establish the Bloom filter as a practical alternative to conventional associative data structures used in computational linguistics. The framework presented here shows that with some consideration for its workings, the randomised nature of the Bloom filter need not be a significant impediment to is use in applications.

Acknowledgements

References

T.C. Bell, J.G. Cleary, and I.H. Witten. 1990. *Text Compression*. Prentice Hall, Englewood Cliffs, NJ.

B. Bloom. 1970. Space/time tradeoffs in hash coding with allowable errors. *CACM*, 13:422–426.

A. Broder and M. Mitzenmacher. 2005. Network applications of Bloom filters: A survey. *Internet Mathematics*, 1(4):485–509.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

J. Goodman and J. Gao. 2000. Language model size reduction by pruning and clustering. In *ICSLP'00*, Beijing, China.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP/Co-NLL)*.

P. Koehn. 2003. Europarl: A multilingual corpus for evaluation of machine translation, draft. Available at:http://people.csail.mit.edu/ koehn/publications/europarl.ps.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL-2002: 40th Annual meeting of the Association for Computational Linguistics*.

Andreas Stolcke. 1998. Entropy-based pruning of back-off language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*.

D. Talbot and M. Osborne. 2007. Randomised language modelling for statistical machine translation. In *45th Annual Meeting of the Association of Computational Linguists (To appear)*.

E. Whitaker and B. Raj. 2001. Quantization-based language model compression (tr-2001-41). Technical report, Mitsubishi Electronic Research Laboratories.

Word Sense Disambiguation Incorporating Lexical and Structural Semantic Information

Takaaki Tanaka[†] **Francis Bond**[◇] **Timothy Baldwin**[♠] **Sanae Fujita**[†] **Chikara Hashimoto**[♣]
[†] {takaaki, sanae}@cslab.kecl.ntt.co.jp [◇] bond@nict.go.jp
[♠] tim@csse.unimelb.edu.au [♣] ch@yz.yamagata-u.ac.jp

[†] NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation
[◇] National Institute of Information and Communications Technology
[♠] The University of Melbourne [♣] Yamagata University

Abstract

We present results that show that incorporating lexical and structural semantic information is effective for word sense disambiguation. We evaluated the method by using precise information from a large treebank and an ontology automatically created from dictionary sentences. Exploiting rich semantic and structural information improves precision 2–3%. The most gains are seen with verbs, with an improvement of 5.7% over a model using only bag of words and n-gram features.

1 Introduction

Recently, significant improvements have been made in combining symbolic and statistical approaches to various natural language processing tasks. In parsing, for example, symbolic grammars are being combined with stochastic models (Riezler et al., 2002; Oepen et al., 2002; Malouf and van Noord, 2004). Statistical techniques have also been shown to be useful for word sense disambiguation (Stevenson, 2003). However, to date, there have been few combinations of sense information together with symbolic grammars and statistical models. Klein and Manning (2003) show that much of the gain in statistical parsing using lexicalized models comes from the use of a small set of function words. Features based on general relations provide little improvement, presumably because the data is too sparse: in the Penn treebank normally used to train and test statistical parsers *stocks* and *skyrocket* never appear together. They note that this should motivate the use of similarity and/or class based approaches: the superordinate concepts *capital* ($\supset$ *stocks*) and *move upward* ($\supset$ *sky rocket*) frequently appear together. However, there has been little success in this area to date. For example, Xiong et al. (2005) use semantic knowledge to parse Chinese, but gain only a marginal improvement. Focusing on WSD, Stevenson (2003) and others have shown that the use of syntactic information (predicate-argument relations) improve the quality of word sense disambiguation (WSD). McCarthy and Carroll (2003) have shown the effectiveness of the selectional preference information for WSD. However, there is still little work on combining WSD and parse selection.

We hypothesize that one of the reasons for the lack of success is that there has been no resource annotated with both syntactic (or structural semantic information) and lexical semantic information. For English, there is the SemCor corpus (Fellbaum, 1998) which is annotated with parse trees and WordNet senses, but it is fairly small, and does not explicitly include any structural semantic information. Therefore, we decided to construct and use a treebank with both syntactic information (e.g. HPSG parses) and lexical semantic information (e.g. sense tags): the Hinoki treebank (Bond et al., 2004). This can be used to train word sense disambiguation and parse ranking models using both syntactic and lexical semantic features. In this paper, we discuss only word sense disambiguation. Parse ranking is discussed in Fujita et al. (2007).

2 The Hinoki Corpus

The Hinoki corpus consists of the Lexeed Semantic Database of Japanese (Kasahara et al., 2004) and corpora annotated with syntactic and semantic infor-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 477–485, Prague, June 2007. ©2007 Association for Computational Linguistics

mation.

2.1 Lexeed

Lexeed is a database built from on a dictionary, which defines word senses used in the Hinoki corpus and has around 49,000 dictionary definition sentences and 46,000 example sentences which are syntactically and semantically annotated. Lexeed consists of all words with a familiarity greater than or equal to five on a scale of one to seven. This gives a fundamental vocabulary of 28,000 words, divided into 46,347 different senses. Each sense has a definition sentence and example sentence written using only these 28,000 familiar words (and some function words). Many senses have more than one sentence in the definition: there are 75,000 defining sentences in all.

A (simplified) example of the entry for 運転手 *untenshu* "chauffeur" is given in Figure 1. Each word contains the word itself, its part of speech (POS) and lexical type(s) in the grammar, and the familiarity score. Each sense then contains definition and example sentences, links to other senses in the lexicon (such as hypernym), and links to other resources, such as the Goi-Taikei (Ikehara et al., 1997) and WordNet (Fellbaum, 1998). Each content word in the definition and example sentences is annotated with sense tags from the same lexicon.

2.2 Lexical Semantics Annotation

The lexical semantic annotation uses the sense inventory from Lexeed. All words in the fundamental vocabulary are tagged with their sense. For example, the word 大きい *ookii* "big" (in *ookiku naru* "grow up") is tagged as sense 5 in the example sentence (Figure 1), with the meaning "elder, older".

Each word was annotated by five annotators. We use the majority choice in case of disagreements (Tanaka et al., 2006). Inter-annotator agreements among the five annotators range from 78.7% to 83.3%: the lowest agreement is for the Lexeed definition sentences and the highest is for Kyoto corpus (newspaper text). These agreements reflect the difficulties in disambiguating word sense over each corpus and can be considered as the upper bound of precision for WSD.

Table 1 shows the distribution of word senses according to the word familiarity in Lexeed.

		Poly-		#Mono-
Fam	#Words	semous	#WS	semous(%)
6.5 -	368	182	4.0	186 (50.5)
6.0 -	4,445	1,902	3.4	2,543 (57.2)
5.5 -	9,814	3,502	2.7	6,312 (64.3)
5.0 -	11,430	3,457	2.5	7,973 (69.8)

Table 1: Word Senses in Lexeed

2.3 Ontology

The Hinoki corpus comes with an ontology semi-automatically constructed from the parse results of definitions in Lexeed (Nichols and Bond, 2005). The ontology includes more than 80 thousand relationships between word senses, e.g. synonym, hypernym, abbreviation, etc. The hypernym relation for 運転手 *untenshu* "chauffeur" is shown in Figure 1. Hypernym or synonym relations exist for almost all content words.

2.4 Thesaurus

As part of the ontology verification, all nominal and most verbal word senses in Lexeed were linked to semantic classes in the Japanese thesaurus, Nihongo Goi-Taikei (Ikehara et al., 1997). These were then hand verified. Goi-Taikei has about 400,000 words including proper nouns, most nouns are classified into about 2,700 semantic classes. These semantic classes are arranged in a hierarchical structure (11 levels). The Goi-Taikei Semantic Class for 運転手 *untenshu* "chauffeur" is shown in Figure 1: $\langle$`C292:driver`$\rangle$ at level 9 which is subordinate to $\langle$`C4:person`$\rangle$.

2.5 Syntactic and Structural Semantics Annotation

Syntactic annotation is done by selecting the best parse (or parses) from the full analyses derived by a broad-coverage precision grammar. The grammar is an HPSG implementation (JACY: Siegel and Bender, 2002), which provides a high level of detail, marking not only dependency and constituent structure but also detailed semantic relations. As the grammar is based on a monostratal theory of grammar (HPSG: Pollard and Sag, 1994) it is possible to simultaneously annotate syntactic and semantic structure without overburdening the annotator. Using a grammar enforces treebank consistency — all sentences annotated are guaranteed to have well-

INDEX	運転手 *untenshu*	
POS	noun	
LEX-TYPE	`noun-lex`	
FAMILIARITY	6.2 [1–7] ($\geq$ 5)	
SENSE 1	DEFINITION	電車$_1$ や 自動車$_1$ を 運転$_1$ する 人$_4$ a person who drives trains and cars
	EXAMPLE	大きく$_5$ なったら電車$_1$ の運転手$_1$ に成る$_6$ のが夢$_3$ です。 I dream of growing up and becoming a train driver
	HYPERNYM	人$_4$ *hito* “person”
	SEM. CLASS	⟨`292:driver`⟩ (⊂ ⟨`4:person`⟩)
	WORDNET	*motorman*$_1$

Figure 1: Dictionary Entry for 運転手$_1$ *untenshu* “chauffeur”

formed parses. The flip side to this is that any sentences which the parser cannot parse remain unannotated, at least unless we were to fall back on full manual mark-up of their analyses. The actual annotation process uses the same tools as the Redwoods treebank of English (Oepen et al., 2002).

There were 4 parses for the definition sentence shown in Figure 1. The correct parse, shown as a phrase structure tree, is shown in Figure 2. The two sources of ambiguity are the conjunction and the relative clause. The parser also allows the conjunction to join to 電車 *densha* and 人 *hito*. In Japanese, relative clauses can have gapped and non-gapped readings. In the gapped reading (selected here), 人 *hito* is the subject of 運転 *unten* “drive”. In the non-gapped reading there is some underspecified relation between the thing and the verb phrase. This is similar to the difference in the two readings of *the day he knew* in English: “the day that he knew about” (gapped) vs “the day on which he knew (something)” (non-gapped). Such semantic ambiguity is resolved by selecting the correct derivation tree that includes the applied rules in building the tree.

The parse results can be automatically given by the HPSG parser PET (Callmeier, 2000) with the Japanese grammar JACY. The current parse ranking model has an accuracy of 70%: the correct tree is ranked first 70% of the time (for Lexeed definition sentences) (Fujita et al., 2007).

The full parse is an HPSG sign, containing both syntactic and semantic information. A view of the semantic information is given in Figure 3[1].

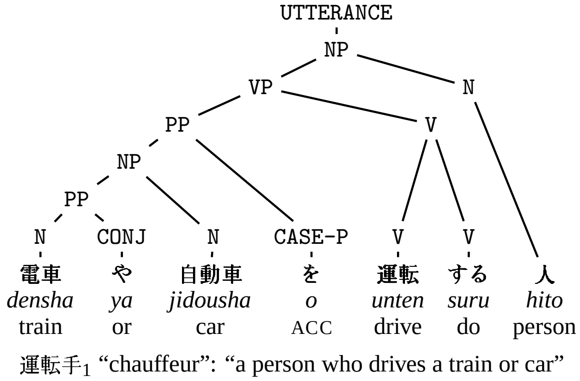


Figure 2: Syntactic View of the Definition of 運転手$_1$ *untenshu* “chauffeur”

The semantic view shows some ambiguity has been resolved that is not visible in the purely syntactic view.

The semantic view can be further simplified into a dependency representation, further abstracting away from quantification, as shown in Figure 4. One of the advantages of the HPSG sign is that it contains all this information, making it possible to extract the particular view needed. In order to make linking to other resources (such as the sense annotation) easier, predicates are labeled with pointers back to their position in the original surface string. For example, the predicate `densha_n_1` links to the surface characters between positions 0 and 3: 電車.

[1] The specific meaning representation language used in JACY is Minimal Recursion Semantics (Copestake et al., 2005).

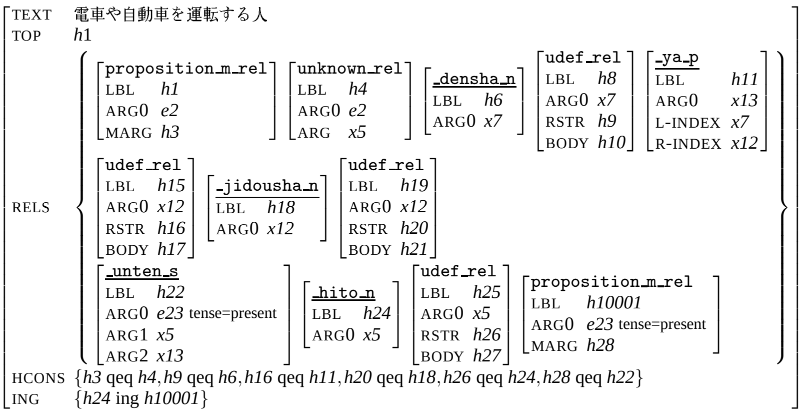

TEXT 電車や自動車を運転する人
TOP h1
RELS
proposition_m_rel LBL h1 ARG0 e2 MARG h3
unknown_rel LBL h4 ARG0 e2 ARG x5
_densha_n LBL h6 ARG0 x7
udef_rel LBL h8 ARG0 x7 RSTR h9 BODY h10
_ya_p LBL h11 ARG0 x13 L-INDEX x7 R-INDEX x12
udef_rel LBL h15 ARG0 x12 RSTR h16 BODY h17
_jidousha_n LBL h18 ARG0 x12
udef_rel LBL h19 ARG0 x12 RSTR h20 BODY h21
_unten_s LBL h22 ARG0 e23 tense=present ARG1 x5 ARG2 x13
_hito_n LBL h24 ARG0 x5
udef_rel LBL h25 ARG0 x5 RSTR h26 BODY h27
proposition_m_rel LBL h10001 ARG0 e23 tense=present MARG h28
HCONS {h3 qeq h4, h9 qeq h6, h16 qeq h11, h20 qeq h18, h26 qeq h24, h28 qeq h22}
ING {h24 ing h10001}

Figure 3: Semantic View of the Definition of 運転手$_1$ *untenshu* “chauffeur”

```
_1:proposition_m<0:13>[MARG e2:unknown]
e2:unknown<0:13>[ARG x5:_hito_n]
x7:udef<0:3>[]
x7:densha_n_1<0:3>
x12:udef<4:7>[]
x12:_jidousha_n<4:7>
x13:_ya_p_conj<0:4>[L-INDEX x7:_densha_n_1, R-INDEX x12:_jidousha_n]
e23:_unten_s_2<8:10>[ARG1 x5:_hito_n, ARG2 x13:_ya_p_conj]
x5:udef<12:13>[]
_2:proposition_m<0:13>[MARG e23:_unten_s_2]
```

Figure 4: Dependency View of the Definition of 運転手$_1$ *untenshu* “chauffeur”

3 Task

We define the task in this paper as “allocating the word sense tags for all content words included in Lexeed as headwords, in each input sentence”. This task is a kind of all-words task, however, a unique point is that we focus on fundamental vocabulary (basic words) in Lexeed and ignore other words. We use Lexeed as the sense inventory. There are two problems in resolving the task: how to build the model and how to assign the word sense by using the model for disambiguating the senses. We describe the word sense selection model we use in section 4 and the method of word sense assignment in section 5.

4 Word Sense Selection Model

All content words (i.e. basic words) in Lexeed are classified into six groups by part-of-speech: noun, verb, verbal noun, adjective, adverb, others. We treat the first five groups as targets of disambiguating senses. We build five words sense models corresponding to these groups. A model contains senses for various words, however, features for a word are discriminated from those for other words so that the senses irrelevant to a target word are not selected. For example, an n-gram feature following a target word “has-a-tail” for *dog* is distinct from that for *cat*.

In the remainder of this section, we describe the features used in the word sense disambiguation. First we used simple n-gram collocations, then a bag of words of all words occurring in the sentence. This was then enhanced by using ontological information and predicate argument relations.

4.1 Word Collocations

Word collocations (WORD-Col) are basic and effective cues for WSD. They can be modelled by n-gram and bag of words features, which are easily extracted from a corpus. We used all unigrams, bigrams and trigrams which precede and follow the target words (N-gram) and all content words in the sentences where the target words occur (BOW).

#	sample features
C1	⟨COLWS:人$_4$⟩
C2	⟨COLWS$_{SC}$:C33:other person⟩
C3	⟨COLWS$_{HYP}$:人間$_1$⟩
C4	⟨COLWS$_{HYPSC}$:C5:person⟩
C1	⟨COLWS:電車$_1$⟩
C2	⟨COLWS$_{SC}$:C988:land vehicle⟩
C3	⟨COLWS$_{HYP}$:車両$_1$⟩
C4	⟨COLWS$_{HYPSC}$:C988:land vehicle⟩
C1	⟨COLWS:自動車$_1$⟩
C2	⟨COLWS$_{SC}$:C988:land vehicle⟩
C3	⟨COLWS$_{HYP}$:車$_2$⟩
C4	⟨COLWS$_{HYPSC}$:C988:land vehicle⟩

Table 2: Example semantic collocation features (SEM-Col) extracted from the word sense tagged corpus and the dictionary (Lexeed and GoiTaikei) and the ontology which have the word senses and the semantic classes linked to the semantic tags. The first column numbers the feature template corresponding to each example.

4.2 Semantic Features

We use the semantic information (sense tags and ontologies) in two ways. One is to enhance the collocations and the other is to enhance dependency relations.

4.2.1 Semantic Collocations

Word surface features like N-gram and BOW inevitably suffer from data sparseness, therefore, we generalize them to more abstract words or concepts and also consider words having the same meanings. We used the ontology described in Section 2.3 to get hypernyms and synonyms and the Goi-Taikei thesaurus to abstract the words to the semantic classes. The superordinate classes at level 3, 4 and 5 are also added in addition to the original semantic class. For example, 電車 *densha* "train" and 自動車 *jidousha* "automobile" are both generalized to the semantic class ⟨C988:land vehicle⟩ (level 7). The superordinate classes are also used: ⟨C706:inanimate⟩ (level 3), ⟨C760:artifact⟩ (level 4) and ⟨C986:vehicle⟩ (level 5).

4.2.2 Semantic Dependencies

The semantic dependency features are based on a predicate and its arguments taken from the elementary dependencies. For example, consider the semantic dependency representation for *densha ya jidousha-wo unten suru hito* "a person who drives a train or car" given in Figure 4. The predicate *unten* "drive", has two arguments: ARG1 *hito* "person" and ARG2 *ya* "or". The coordinate conjunction is expanded out into its children, giving ARG2 *densha* "train" and *jidousha* "automobile".

#	sample features for 運転する$_1$
D1	⟨PRED:運転する, ARG1:人⟩
D1	⟨PRED:運転する, ARG2:電車⟩
D1	⟨PRED:運転する, ARG2:自動車⟩
D2	⟨PRED:運転する, ARG1:人$_4$⟩
D2	⟨PRED:運転する, ARG2:電車$_1$⟩
D2	⟨PRED:運転する, ARG2:自動車$_1$⟩
D3	⟨PRED:運転する, ARG1$_{SC}$:C33⟩
D3	⟨PRED:運転する, ARG2$_{SC}$:C988⟩
D4	⟨PRED:運転する, ARG2$_{SYN}$:モーターカー$_1$⟩
D5	⟨PRED:運転する, ARG1$_{HYP}$:人間$_1$⟩
D5	⟨PRED:運転する, ARG2$_{HYP}$:車両$_1$⟩
D5	⟨PRED:運転する, ARG2$_{HYP}$:車$_2$⟩
D6	⟨PRED:運転する, ARG1$_{HYPSC}$:C5⟩
D6	⟨PRED:運転する, ARG2$_{HYPSC}$:C988⟩
D11	⟨PRED:運転する, ARG1:人, ARG2:電車⟩
D22	⟨PRED:運転する, ARG1:人$_4$, ARG2:電車$_1$⟩
D23	⟨PRED:運転する, ARG1:人$_4$, ARG2:C1460 ⟩
D24	⟨PRED:運転する, ARG1:人$_4$, ARG2$_{SYN}$:モーターカー$_1$⟩
D32	⟨PRED:運転する, ARG1:C5, ARG2:電車$_1$⟩
D33	⟨PRED:運転する, ARG1:C5, ARG2:C988⟩
D55	⟨PRED:運転する, ARG1$_{HYP}$:人間$_4$, ARG2$_{HYP}$:車両$_1$⟩
D56	⟨PRED:運転する, ARG1$_{HYP}$:人間$_4$, ARG2$_{HYPSC}$:C988⟩
D65	⟨PRED:運転する, ARG1$_{HYPSC}$:C5 , ARG2$_{HYP}$:車両$_1$⟩
D322	⟨PRED:C2003, ARG1:人$_4$, ARG2:電車$_1$⟩

Table 3: Example semantic features extracted from the dependency tree in Figure 4. The first column numbers the feature template corresponding to each example.

From these, we produce several features, a sample of them are shown in Table 3. One has all arguments and their labels (D11). We also produce various back offs, for example the predicate with only one argument at a time (D1-D3). Each combination of predicate and its related argument(s) becomes a feature.

For the next class of features, we used the sense information from the corpus combined with the semantic classes in the dictionary to replace each predicate by its disambiguated sense, its hypernym, its synonym (if any) and its semantic class. The semantic classes for 電車$_1$ and 自動車$_1$ are both ⟨`988:land vehicle`⟩, while 運転$_1$ is ⟨`2003:motion`⟩ and 人$_4$ is ⟨`4:human`⟩. We also expand 自動車$_1$ into its synonym モーターカー$_1$ *mōtākā* "motor car".

The semantic class features provide a semantic smoothing, as words are binned into the 2,700 classes. The hypernym/synonym features provide even more smoothing. Both have the effect of making more training data available for the disambiguator.

4.3 Domain

Domain information is a simple and sometimes strong cue for disambiguating the target words (Gliozzo et al., 2005). For instance, the sense of the word "*record*" is likey to be different in the musical context, which is recalled by domain-specific words like "*orchestra*", "*guitar*", than in the sporting context. We use 12 domain categories like "culture/art", "sport", etc. which are similar to ones used in directory search web sites. About 6,000 words are automatically classified into one of 12 domain categories by distributions in web sites (Hashimoto and Kurohashi, 2007) and 10% of them are manually checked. Polysemous words which belong to multiple domains and neutral words are not classified into any domain.

5 Search Algorithm

The conditional probability of the word sense for each word is given by the word sense selection model described in Section 4. In the initial state, some of the semantic features, e.g. semantic collocations (SEM-Col) and word sense extensions for semantic dependencies (SEM-Dep) are not available, since no word senses for polysemous words have been determined. It is not practical to count all combinations of word senses for target words, therefore, we first try to decide the sense for that word which is most plausible among all the ambiguous words, then, disambiguate the next word by using the sense.

We use the beam search algorithm, which is similar to that used for decoder in statistical machine translation (Watanabe, 2004), for finding the plausible combination of word sense tags.

The algorithm is described as follows. For a polysemous word set in an input sentence $\{w_1,\ldots,w_n\}$, t_{w_ik} is the k-th word sense of word w_i, W is a set having words to be disambiguated, T is a list of resolved word senses. A search node N is defined as $[W,T]$ and a score of a node N, $s(N)$ is defined as the probability that the word sense set T occurs in the context. The beam search can be done as follows (beam width is b):

1. Create an initial node $N_0 = [T_0,W_0]$ ($T_0 = \{\}$, $W_0 = \{\}$) and insert the node into an initial queue Q_0.

2. For each node N in the queue Q, do the following steps.
   - For each w_i $(\in W)$, create W'_i by picking out w_i from W
   - Create new lists $T'_1,\ldots,T'_l$ by adding one of word sense candidates $t_{w_i1},\ldots,t_{w_il}$ for w_i to T
   - Create new nodes $[W'_i,T'_0],\ldots,[W'_i,T'_l]$ and insert them into the queue Q'

3. Sort the nodes in Q' by the score $s(N)$

4. If the top node W in the queue Q' is empty, adopt T as the combination of word senses and terminate. Otherwise, pick out the top b nodes from Q' and insert them into new queue Q, then go back to 2

6 Evaluation

We trained and tested on the Lexeed Dictionary Definition (LXD-DEF) and Example sections (LXD-EX) of the Hinoki corpus (Bond et al., 2007). These have about 75,000 definition and 46,000 example sentences respectively. Some 54,000 and 36,000 sentences of them are treebanked, i.e., they have the syntactic trees and structural semantic information. We used these sentences with the complete information and selected 1,000 sentences out of each sentence class as test sets (LXD-DEF$_{\text{test}}$, LXD-EX$_{\text{test}}$), and the remainder is combined and used as a training set (LXD-ALL). We also tested 1,000 sentences from the Kyoto Corpus of newspaper text (KYOTO$_{\text{test}}$). These sentences have between 3.4 (LXD-EX$_{\text{test}}$) – 5.2 (KYOTO$_{\text{test}}$) polysemous words per sentence on average.

We use a *maximum entropy / minimum divergence* (MEMD) modeler to train the word sense selection model. We use the open-source Maximun Entropy Modeling Toolkit[2] for training, determining best-performing convergence thresholds and prior sizes experimentally. The models for five different POSs were trained with each training sets: the base model is word collocation model (WORD-Col), and the semantic models built by semantic collocation (SEM-Col), semantic dependency (SEM-Dep) or domain with WORD-Col (+SEM-Col, +SEM-Dep and +DOMAIN).

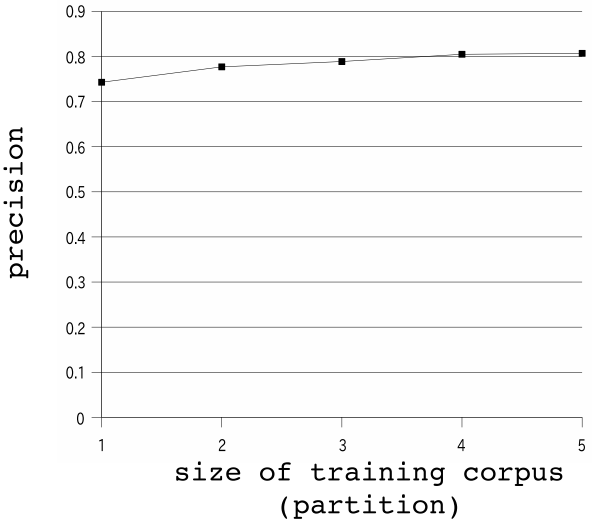


Figure 5: Learning Curve

7 Results and Discussion

Table 4 shows the precision as the results of the word sense disambiguation on the combination of LXD-DEF and LXD-EX (LXD-ALL). The baseline method selects the senses occurring most frequently in the training corpus. Each row indicates the results using the baseline, word collocation (WORD-Col), the combinations of WORD-Col and one of the semantic features (+SEM-Col, +SEM-Dep and +DOMAIN), e.g, +SEM-Col gives the results using WORD-Col and SEM-Col, and all features (FULL).

There are significant improvements over the baseline and the other results on all corpora. Basic word collocation features (WORD-Col) give a vast improvement. Extending this by using the ontological information (+SEM-Col) gives a further improvement over the WORD-Col. Adding the predicate-argument relationships (+SEM-Dep) improves the results even more.

Table 6 shows the statistics of the target corpora. The best result of LXD-DEF$_{test}$ (80.7%) surpasses the inter-annotator agreement (78.7%) in building the Hinoki Sensebank. However, there is a wide gap between the best results of KYOTO$_{test}$ (60.4%) and the inter-annotator agreement (83.3%), this suggests other information such as the semantic classes for named entities (including proper nouns and multiword expressions (MWE)) and broader contexts are required. However, a model built on dictionary sentences lacks these features. Even, so there is some improvement.

The domain features (+DOMAIN) give small contribution to the precision, since only intra-sentence context is counted in this experiment. Unfortunately dictiory definition and example sentences do not really have a useful context. We expect broader context should make the domain features more effective for the newspaper text (e.g. as in Stevenson (2003)),

Table 5 shows comparison of results of different POSs. The semantic features (+SEM-Col and +SEM-Dep) are particularly effective for verb and also give moderate improvements on the results of the other POSs.

Figure 5 shows the precisions of LXD-DEF$_{test}$ in changing the size of a training corpus, which is divided into five partitions. The precision is saturated in using four partitions (264,000 tokens).

These results of the dictionary sentences are close to the best published results for the SENSEVAL-2 task (79.3% by Murata et al. (2003) using a combination of simple Bayes learners). However, we are using a different sense inventory (Lexeed not Iwanami (Nishio et al., 1994)) and testing over a different corpus, so the results are not directly comparable. In future work, we will test over SENSEVAL-2 data so that we can compare directly.

None of the SENSEVAL-2 systems used ontological information, despite the fact that the dictionary definition sentences were made available, and there are several algorithms describing how to extract such information from MRDs (Tsurumaru

[2] `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`

Model	Test	Baseline	WORD-Col	+SEM-Col	+SEM-Dep	+DOMAIN	FULL
LXD-ALL	LXD-DEF$_{\text{test}}$	72.8	78.4	79.8	80.2	78.1	80.7
	LXD-EX$_{\text{test}}$	70.4	75.6	78.7	77.9	76.0	78.8
	KYOTO$_{\text{test}}$	55.6	58.5	60.0	58.8	59.8	60.4

Table 4: The Precision of WSD

POS	Baseline	WORD-Col	+SEM-Col	+SEM-Dep	+DOMAIN	FULL
Noun	65.5	68.7	69.6	69.4	68.9	69.8
Verb	60.3	66.9	71.0	70.6	67.7	72.6
VN	72.6	76.2	77.7	74.6	77.6	77.5
Adj	59.9	67.2	69.5	68.9	68.9	69.5
Adv	74.4	78.6	79.8	79.2	78.6	79.8

Table 5: The Precision of WSD (per Part-of-Speech)

et al., 1991; Wilkes et al., 1996; Nichols et al., 2005). We hypothesize that this is partly due to the way the task is presented: there was not enough time to extract and debug an ontology as well as build a disambiguation system, and there was no ontology distributed. The CRL system (Murata et al., 2003) used a syntactic dependency parser as one source of features (KNP: Kurohashi and Nagao (2003)), removing it decreased performance by around 0.6%.

8 Conclusions

We used the Hinoki corpus to test the importance of lexical and structural information in word sense disambiguation. We found that basic n-gram features and collocations provided a great deal of useful information, but that better results could be gained by using ontological information and semantic dependencies.

Acknowledgements

We would like to thank the other members of the NTT Natural Language Research Group NTT Communication Science laboratories for their support. We would also like to express gratitude to the reviewers for their valuable comments and Professor Zeng Guangping, Wang Daliang and Shen Bin of the University of Science and Technology Beijing (USTB) for building the demo system.

References

Francis Bond, Sanae Fujita, Chikara Hashimoto, Kaname Kasahara, Shigeko Nariyama, Eric Nichols, Akira Ohtani, Takaaki Tanaka, and Shigeaki Amano. 2004. The Hinoki treebank: A treebank for text understanding. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 554–559. Hainan Island.

Francis Bond, Sanae Fujita, and Takaaki Tanaka. 2007. The Hinoki syntactic and semantic treebank of Japanese. *Language Resources and Evaluation*. (Special issue on Asian language technology).

Ulrich Callmeier. 2000. PET - a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–108.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics. An introduction. *Research on Language and Computation*, 3(4):281–332.

Christine Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Sanae Fujita, Francis Bond, Stephan Oepen, and Takaaki Tanaka. 2007. Exploiting semantic information for HPSG parse selection. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 25–32. Prague, Czech Republic.

Alfio Massimiliano Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*. Ann Arbor, U.S.

Chikara Hashimoto and Sadao Kurohashi. 2007. Construction of domain dictionary for fundamental vocaburalry. In *Proceedings of the ACL 2007 Main Conference Poster Sessions*. Association for Computational Linguistics, Prague, Czech Republic.

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, and Yoshihiko Hayashi. 1997. *Goi-Taikei — A Japanese Lexicon*. Iwanami Shoten, Tokyo. 5 volumes/CDROM.

Kaname Kasahara, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kanasugi, and Shigeaki Amano. 2004. Construction of a Japanese semantic lexicon: Lexeed. In *IPSG SIG: 2004-NLC-159*, pages 75–82. Tokyo. (in Japanese).

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430. URL `http://www.aclweb.org/anthology/P03-1054.pdf`.

	Annotated		Agreement				
Corpus	Tokens	#WS	token (type)	%Other Sense	%Homonym	%MWE	%Proper Noun
LXD-DEF	199,268	5.18	.787 (.850)	4.2	0.084	1.5	0.046
LXD-EX	126,966	5.00	.820 (.871)	2.3	0.035	0.4	0.0018
KYOTO	268,597	3.93	.833 (.828)	9.8	3.3	7.9	5.5

Table 6: Corpus Statistics

Sadao Kurohashi and Makoto Nagao. 2003. Building a Japanese parsed corpus — while improving the parsing system. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, chapter 14, pages 249–260. Kluwer Academic Publishers.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*. JST CREST. URL `http://www-tsujii.is.s.u-tokyo.ac.jp/bsa/papers/malouf.pdf`.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

Masaaki Murata, Masao Utiyama, Kiyotaka Uchimoto, Qing Ma, and HItoshi Isahara. 2003. CRL at Japanese dictionary-based task of SENSEVAL-2. *Journal of Natural Language Processing*, 10(3):115–143. (in Japanese).

Eric Nichols and Francis Bond. 2005. Acquiring ontologies using deep and shallow processing. In *11th Annual Meeting of the Association for Natural Language Processing*, pages 494–498. Takamatsu.

Eric Nichols, Francis Bond, and Daniel Flickinger. 2005. Robust ontology acquisition from machine-readable dictionaries. In *Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-2005*, pages 1111–1116. Edinburgh.

Minoru Nishio, Etsutaro Iwabuchi, and Shizuo Mizutani. 1994. *Iwanami Kokugo Jiten Dai Go Han [Iwanami Japanese Dictionary Edition 5]*. Iwanami Shoten, Tokyo. (in Japanese).

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christoper D. Manning, Dan Flickinger, and Thorsten Brant. 2002. The LinGO redwoods treebank: Motivation and preliminary applications. In *19th International Conference on Computational Linguistics: COLING-2002*, pages 1253–7. Taipei, Taiwan.

Carl Pollard and Ivan A. Sag. 1994. *Head Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *41st Annual Meeting of the Association for Computational Linguistics: ACL-2003*, pages 271–278.

Melanie Siegel and Emily M. Bender. 2002. Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics*, pages 1–8. Taipei.

Mark Stevenson. 2003. *Word Sense Disambiguation*. CSLI Publications.

Takaaki Tanaka, Francis Bond, and Sanae Fujita. 2006. The Hinoki sensebank — a large-scale word sense tagged corpus of Japanese —. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006*, pages 62–69. Sydney. URL `http://www.aclweb.org/anthology/W/W06/W06-0608`, (ACL Workshop).

Hiroaki Tsurumaru, Katsunori Takesita, Itami Katsuki, Toshihide Yanagawa, and Sho Yoshida. 1991. An approach to thesaurus construction from Japanese language dictionary. In *IPSJ SIGNotes Natural Language*, volume 83-16, pages 121–128. (in Japanese).

Taro Watanabe. 2004. *Example-based Statistical Machine Translation*. Ph.D. thesis, Kyoto University.

Yorick A. Wilkes, Brian M. Slator, and Louise M. Guthrie. 1996. *Electric Words*. MIT Press.

Deyi Xiong, Qun Liu Shuanglong Li and, Shouxun Lin, and Yueliang Qian. 2005. Parsing the Penn Chinese treebank with semantic knowledge. In Robert Dale, Jian Su Kam-Fai Wong and, and Oi Yee Kwong, editors, *Natural Language Processing — IJCNLP 005: Second International Joint Conference Proceedings*, pages 70–81. Springer-Verlag.

An Approach to Text Corpus Construction which Cuts Annotation Costs and Maintains Reusability of Annotated Data

Katrin Tomanek **Joachim Wermter** **Udo Hahn**
Jena University Language & Information Engineering (JULIE) Lab
Fürstengraben 30
D-07743 Jena, Germany
{tomanek|wermter|hahn}@coling-uni-jena.de

Abstract

We consider the impact Active Learning (AL) has on effective and efficient text corpus annotation, and report on reduction rates for annotation efforts ranging up until 72%. We also address the issue whether a corpus annotated by means of AL – using a particular classifier and a particular feature set – can be re-used to train classifiers different from the ones employed by AL, supplying alternative feature sets as well. We, finally, report on our experience with the AL paradigm under real-world conditions, i.e., the annotation of large-scale document corpora for the life sciences.

1 Introduction

The annotation of corpora has become a crucial prerequisite for NLP utilities which rely on (semi-) supervised machine learning (ML) techniques. While stability, by and large, has been reached for tagsets up until the syntax layer, semantic annotations in terms of (named) entities, semantic roles, propositions, events, etc. reveal a high degree of variability due to the inherent domain-dependence of the underlying tagsets. This diversity fuels a continuous need for creating semantic annotation data anew.

Accordingly, annotation activities will persist and even increase in number as HLT is expanding on various technical and scientific domains (e.g., the life sciences) outside the classical general-language newspaper genre. Since the provision of annotations is a costly, labor-intensive and error-prone process the amount of work and time this activity requires should be minimized to the extent that corpus data could still be used to effectively train ML-based NLP components on them. The approach we advocate does exactly this and yields reduction gains (compared with standard procedures) ranging between 48% to 72%, without seriously sacrificing annotation quality.

Various techniques to minimize the necessary amount of annotated training material have already been investigated. In co-training (Blum and Mitchell, 1998), e.g., from a small initial set of labeled data multiple learners mutually provide new training material for each other by labeling unseen examples. Pierce and Cardie (2001) have shown, however, that for tasks which require large numbers of labeled examples – such as most NLP tasks – co-training might be inadequate because it tends to generate noisy data. Furthermore, a well compiled initial training set is a crucial prerequisite for successful co-training. As another alternative for minimizing annotation work, active learning (AL) is based on the idea to let the learner have control over the examples to be manually labeled so as to optimize the prediction accuracy. Accordingly, AL aims at selecting those examples with high utility for the model.

AL (as well as semi-supervised methods) is typically considered as a learning protocol, i.e., to train a particular classifier. In contrast, we here propose to employ AL as a corpus annotation method. A corpus built on these premises must, however, still be reusable in a flexible way so that, e.g., training with modified or improved classifiers is feasible and reasonable on AL-generated corpora. Baldridge and Osborne (2004) have already argued that this is a highly critical requirement because the examples selected by AL are tuned to one particular classifier. The second major contribution of this paper ad-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 486–495, Prague, June 2007. ©2007 Association for Computational Linguistics

dresses this issue and provides empirical evidence that corpora built with one type of classifier (based on Maximum Entropy) can reasonably be reused by another, methodologically related type of classifier (based on Conditional Random Fields) without requiring changes of the corpus data. We also show that feature sets being used for training classifiers can be enhanced without invalidating corpus annotations generated on the basis of AL and, hence, with a poorer feature set.

2 Related Work

There are mainly two methodological strands of AL research, *viz.* optimization approaches which aim at selecting those examples that optimize some (algorithm-dependent) objective function, such as prediction variance (Cohn et al., 1996), and heuristic methods with uncertainty sampling (Lewis and Catlett, 1994) and query-by-committee (QBC) (Seung et al., 1992) just to name the most prominent ones. AL has already been applied to several NLP tasks, such as document classification (Schohn and Cohn, 2000), POS tagging (Engelson and Dagan, 1996), chunking (Ngai and Yarowsky, 2000), statistical parsing (Thompson et al., 1999; Hwa, 2000), and information extraction (Lewis and Catlett, 1994; Thompson et al., 1999).

In a more recent study, Shen et al. (2004) consider AL for entity recognition based on Support Vector Machines. Here, the informativeness of an example is estimated by the distance to the hyperplane of the currently learned SVM. It is assumed that an example which lies close to the hyperplane has high chances to have an effect on training. This approach is essentially limited to the SVM learning scheme as it solely relies on SVM-internal selection criteria.

Hachey et al. (2005) propose a committee-based AL approach where the committee's classifiers constitute multiple views on the data by employing different feature subsets. The authors focus on (possible) negative side effects of AL on the annotations. They argue that AL annotations are cognitively more difficult to deal with for the annotators (because of the increased complexity of the selected sentences). Hence, lower annotation quality and higher per-sentence annotation times might be a concern.

There are controversial findings on the reusability of data annotated by means of AL for the problem of parse tree selection. Whereas Hwa (2001) reports positive results, Baldridge and Osborne (2004) argue that AL based on uncertainty sampling may face serious performance degradation when labeled data is reused for training a classifier different from the one employed during AL. For committee-based AL, however, there is a lack of work on reusability. Our experiments of committee-based AL for entity recognition, however, reveal that for this task at least, reusability can be guaranteed to a very large extent.

3 AL for Corpus Annotation - Requirements for Practical Use

AL frameworks for real-world corpus annotation should meet the following requirements:

fast selection time cycles — AL-based corpus annotation is an interactive process in which b sentences are selected by the AL engine for human annotation. Once the annotated data is supplied, the AL engine retrains its underlying classifier(s) on *all* available annotations and then re-classifies all unseen corpus items. After that the most informative (i.e., deviant) b sentences from the set of newly classified data are selected for the next iteration round. In this approach the time needed to select the next examples (which is the idle time of the human annotators) has to be kept at an acceptable limit of a few minutes only. There are various AL strategies which – although they yield theoretically near-optimal sample selection – turn out to be actually impractible for real-world use because of excessively high computation times (cf. Cohn et al. (1996)). Thus, AL-based annotation should be based on a computationally tractable and task-wise feasible and acceptable selection strategy (even if this might imply a suboptimal reduction of annotation costs).

reusability — The examples AL selects for manual annotation are dependent on the model being used, up to a certain extent (Baldridge and Osborne, 2004). During annotation time, however, the best model might not be known and model tuning (especially the choice of features) is typically performed once a training corpus is available. Hence, from a practical point of view, the resulting corpus should be reusable with modified classifiers as well.

adaptive stopping criterion — An explicit and adaptive stopping criterion which is sensitive towards the already achieved level of quality of the annotated corpus is clearly preferred over stopping after an *a priori* fixed number of annotation iterations.

If these requirements, especially the first and the second one, cannot be guaranteed for a specific annotation task one should refrain from using AL. The efficiency of AL-driven annotation (in terms of the time needed to compile high quality training material) might be worse compared to the annotation of randomly (or subjectively) selected examples.

4 Framework for AL-based Named Entity Annotation

For named entity recognition (NER), each change of the application domain requires a more or less profound change of the types of semantic categories (tags) being used for corpus annotation. Hence, one may encounter a lack of training material for various relevant (sub)domains. Once this data is available, however, one might want to modify the features of the final classifier with respect to the specific entity types. Thus, a corpus annotated by means of AL has to provide the flexibility to modify the features of the final classifier.

To meet the requirements from above under the constraints of a real-world annotation task, we decided for QBC-based AL, a *heuristic* AL approach, which is computationally less complex and resource-greedy than *objective function* AL methods (the latter explicitly quantify the differences between the current and an ideal classifier in terms of some objective function). Accordingly, we ruled out uncertainty sampling, another heuristic AL approach, because it was shown before that QBC is more efficient and robust (Freund et al., 1997).

QBC is based on the idea to select those examples for manual annotation on which a committee of classifiers disagree most in their predictions (Engelson and Dagan, 1996). A committee consists of a number of k classifiers of the same type (same learning algorithm, parameters, and features) but trained on different subsets of the training data. QBC-based AL is also iterative. In each AL round the committee's k classifiers are trained on the already annotated data C, then a pool of unannotated data P is predicted with each classifier resulting in n automatically labeled versions of P. These are then compared according to their labels. Those with the highest variance are selected for manual annotation.

4.1 Selection Strategy

In each iteration, a batch of b examples is selected for manual annotation. The informativeness of an example is estimated in terms of the *disagreement*, i.e., the uncertainty among the committee's classifiers on classifying a particular example. This is measured by the *vote entropy* (Engelson and Dagan, 1996), i.e., the entropy of the distribution of classifications assigned to an example by the classifiers. Vote entropy is defined on the token level t as:

$$D_{tok}(t) := -\frac{1}{\log k} \sum_{l_i} \frac{V(l_i,t)}{k} \log \frac{V(l_i,t)}{k}$$

where $\frac{V(l_i,t)}{k}$ is the ratio of k classifiers where the label l_i is assigned to a token t. As (named) entities often span more than a single text token we consider complete sentences as a reasonable example size unit[1] for AL and calculate the disagreement of a sentence D_{sent} as the mean vote entropy of its single tokens. Since the vote entropy is minimal when all classifiers agree in their vote, sentences with high disagreement are preferred for manual annotation. With informed decisions of human annotators made available, the potential for future disagreement of the classifier committee on conflicting instances should decrease. Thus, each AL iteration selects the b sentences with the highest disagreement to focus on the most controversial decision problems.

Besides informativeness, additional criteria can be envisaged for the selection of examples, e.g., *di-*

[1] Sentence-level examples are but one conceivable grain size – lower grains (such as clauses or phrases) as well as higher grains (e.g., paragraphs or abstracts) are equally possible, with different implications for the AL process.

feature class	description
orthographical	based on regular expressions (e.g. *HasDash*, *IsGreek*, ...), token transformation rule: capital letters replaced by “A”, lowercase letters by “a”, digits by “0”, etc. (e.g., *IL2 → AA0*, *have → aaaa*)
lexical and morphological	prefix and suffix of length 3, stemmed version of each token
syntactic	the token’s part-of-speech tag
contextual	features of neighboring tokens

Table 1: Features used for AL

versity of a batch and *representativeness* of the respective example (to avoid outliers) (Shen et al., 2004). We experimented with these more sophisticated selection strategies but preliminary experiments did not reveal any significant improvement of the AL performance. Engelson and Dagan (1996) confirm this observation that, in general, different (and even more refined) selection methods still yield similar results. Moreover, strategies incorporating more selection criteria often require more parameters to be set. However, proper parametrization is hard to achieve in real-world applications. Using disagreement exclusively for selection requires only one parameter, *viz.* the batch size b, to be specified.

4.2 Classifier and Features

For our AL framework we decided to employ a Maximum Entropy (ME) classifier (Berger et al., 1996). We employ a rich set of features (see Table 1) which are general enough to be used in most (sub)domains for entity recognition. We intentionally avoided using features such as semantic triggers or external dictionary look-ups because they depend a lot on the specific subdomain and entity types being used. However, one might add them to fin- tune the final classifier, if needed. ME classifiers outperform their generative counterparts (e.g., Naïve Bayesian classifiers) because they can easily handle overlapping, probably dependent features which might be contained in rich feature sets. We also favored an ME classifier over an SVM one because the latter is computationally much more complex on rich feature sets and multiple classes and is thus not so well suited for an interactive process like AL.

It has been shown that *Conditional Random Fields* (CRF) (Lafferty et al., 2001) achieve higher performance on many NLP tasks, such as NER, but on the other hand they are computionally more complex than an ME classifier making them also impractical for the interactive AL process. Thus, in our committee we employ ME classifiers to meet requirement 1 (fast selection time cycles). However, in the end we want to use the annotated corpora to train a CRF and will thus examine the reusability of such an ME-annotated AL corpus for CRFs (cf. Subsection 5.2).

4.3 Stopping Criterion

A question hardly addressed up until now is when to actually terminate the AL process. Usually, it gets stopped when the supervized learning performance of the specific classifier is achieved. The problem with such an approach is, however, that in practice one does not know the performance level which could possibly be achieved on an unannotated corpus.

An apparent way to monitor the progress of the annotation process is to periodically (e.g., after each AL iteration) train a classifier on the data annotated so far and evaluate it against some randomly selected gold standard. When the relative performance growth of each AL iteration falls below a certain threshold this might be a good reason to stop the annotation. Though this is probably the most reliable way, it is impractical for many scenarios since assembling and manually annotating a representative gold standard may already be quite a laborious task. Thus, a measure from which we can *predict* the development of the learning curve would be beneficial.

One way to achieve this goal is to monitor the rate of disagreement among the different classifiers after each iteration. This rate will descend as the classifiers get more and more robust in their predictions on unseen data. Thus, an average disagreement approaching zero can be interpreted as an indication that additional annotations will not render any further improvement. In our experiments, we will show that this is a valid stopping criterion, indeed.

5 Experiments and Results

For our experiments, we specified the following three parameters: the batch size b (i.e., the number of sentences to be selected for each AL iteration), the size and composition of the initial training set, and the number of k classifiers in a committee. The smaller the batch size, the higher the AL performance turns out to be. In the special case of batch size of $b = 1$ only that example with the highest disagreement is selected. This is certainly impractical since after each AL iteration a new committee of classifiers has to be trained causing unwarranted annotation idle time. We found $b = 20$ to be a good compromise between the annotators' idle time and AL performance. The initial training set also contains 20 sentences which are randomly selected though. Our committee consists of $k = 3$ classifiers, which is a good trade-off between computational complexity and diversity. Although the AL iterations were performed on the sentence level, we report on the number of annotated tokens. Since sentences may considerably vary in their length the number of tokens constitutes a better measure for annotation costs.

We ran our experiments on two common entity-annotated corpora from two different domains (see Table 2). From the general-language newspaper domain, we used the English data set of the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003). It consists of a collection of newswire articles from the Reuters Corpus,[2] which comes annotated with three entity types: *persons*, *locations*, and *organizations*. From the sublanguage biology domain we used the oncology part of the PENNBIOIE corpus which consists of some 1150 PubMed abstracts. Originally, this corpus contains gene, variation event, and malignancy entity annotations. Manual annotation after each AL round was simulated by moving the selected sentences from the pool of unannotated sentences P to the training corpus T. For our simulations, we built two subcorpora by filtering out entity annotations: the PENNBIOIE gene corpus (PBgene), including the three gene entity subtypes *generic*, *protein*, and *rna*, and the PENNBIOIE variation events corpus (PBvar) corpus including the variation entity subtypes *type*, *event*, *location*, *state-altered*, *state-generic*, and *state-original*. We split all three corpora into two subsets, *viz.* AL simulation data and gold standard data on which we evaluate[3] a classifier in terms of f-score trained on the annotated corpus after each AL iteration (learning curve). As far as the CoNLL corpus is concerned, we have used CoNLL's training set for AL and CoNLL's test set as gold standard. As for PBgene and PBvar, we randomly split the corpora into 90% for AL and 10% as gold standard.

corpus	data set	sentences	tokens
CONLL 3 entities	AL	14,040	203,617
	Gold	3,453	46,435
PBGENE 3 entities	AL	10,050	249,490
	Gold	1,114	27,563
PBVAR 6 entities	AL	10,050	249,490
	Gold	1,114	27,563

Table 2: Corpora used in the Experiments

In the following experiments we will refer to the classifiers used in the AL committee as *selectors*, and the classifier used for evaluation as the *tester*.

5.1 Efficiency of AL and the Applicability of the Stopping Criterion

In a first series of experiments, we evaluated whether AL-based annotations can significantly reduce the human effort compared to the standard annotation procedure where sentences are selected randomly (or subjectively). We also show that disagreement is an accurate stopping criterion. As described in Section 4.2, we here employed a committee of ME classifiers for AL; a CRF was used as tester for both the AL and the random selection. Figures 1, 2, and 3 depict the learning curves for AL selection and random selection (upper two curves) and the respective disagreement curves (lower curve). The random selection curves contained in these plots are averaged over three random selection runs.

With AL, we get a maximum f-score of ≈ 84.5% on the CoNLL corpus after about 118,000 tokens. At about the same number of tokens the disagreement curve drops down to values of around $D_{sent} = 0$. Comparing AL and random selection, an f-score of ≈ 84% is reached after 86,000 and 165,000 tokens, respectively, which means a reduction of annotation costs of about 48%. On PBgene, the effect of AL is comparable: a maximum value of 83.5% f-score is reached first after about 124,000 tokens, a data point where hardly any disagreement between the committee's classifiers occurs. For, e.g., an f-score of

[2] `http://trec.nist.gov/`

[3] We use a strict evaluation criterion which only counts exact matches as true positives because annotations having incorrect boundaries are insufficient for manual corpus annotation.

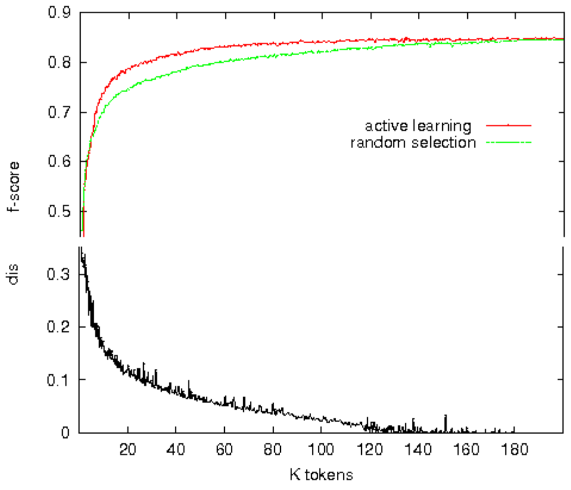


Figure 1: CoNLL Corpus: Learning/Disagreement Curves

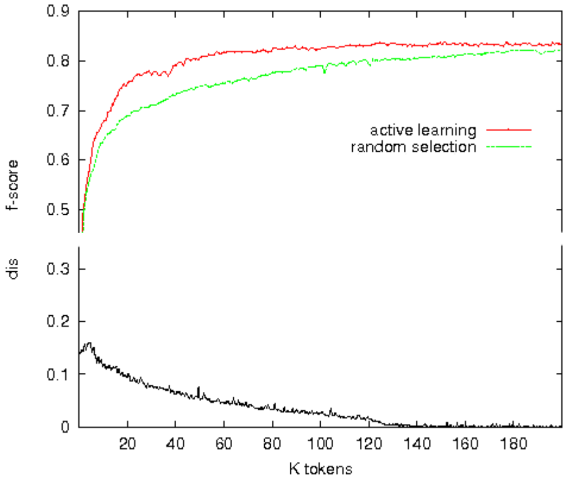


Figure 2: PBgene Corpus: Learning/Disagreement Curves

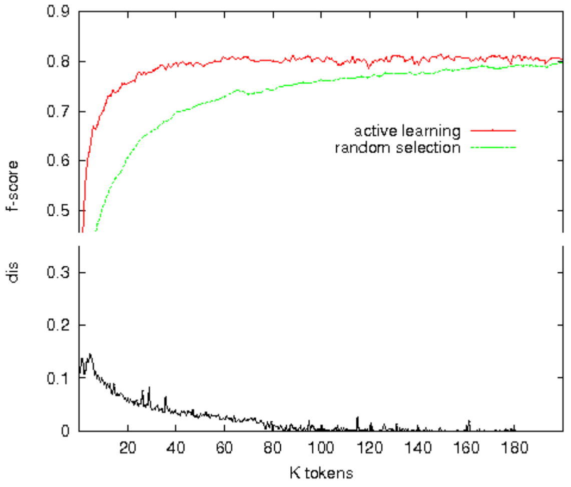


Figure 3: PBvar Corpus: Learning/Disagreement Curves

corpus	selection	F	tokens	reduction
CoNLL	random	84.0	165,000	
	AL	84.0	86,000	≈ 48%
PBgene	random	83.0	101,000	
	AL	83.0	213,000	≈ 53%
PBvar	random	80.0	56,000	
	AL	80.0	200,000	≈ 72%

Table 3: Reduction of Annotation Costs Achieved with AL-based Annotation

83%, the annotation effort can be reduced by about 53% using AL. On PBvar, an f-score of about 80% is reached after ≈ 56,000 tokens when using AL selection, while 200,000 tokens are needed with random selection. For this task, AL reduces the annotation effort by of 72%. Here, the disagreement curve approaches values of zero after approximately 80,000 tokens. At about this point the learning curve reaches its maximum of about 81% f-score. Table 3 summarizes the reduction of annotation costs achieved on all three corpora.

Comparing both PennBioIE simulations, obviously, the reduction of annotation costs through AL is much higher for the variation type entities than for the gene entities. We hypothesize this to be mainly due to incomparable entity densities. Whereas the gene entities are quite frequent (about 1.3 per sentence on average), the variation entities are rather sparse (0.62 per sentence on average) making it an ideal playground for AL-based annotation. Our experiments also reveal that disagreement approaching values of zero is a valid stopping criterion. This is, under all circumstances, definitely the point when AL-based annotation *should* stop because then all classifiers of the committee vote consistently. Any further selection – even though AL selection is used – is then, actually, a *random* selection. If, due to reasons whatsoever, further annotations are wanted, a direct switch to random selection is advisable because this is computationally less expensive than AL-based selection.

5.2 Reusability

To evaluate whether the proposed AL framework for named entity annotation allows for flexible re-use of the annotated data, we performed experiments where we varied both the learning algorithms and the features of the selectors.

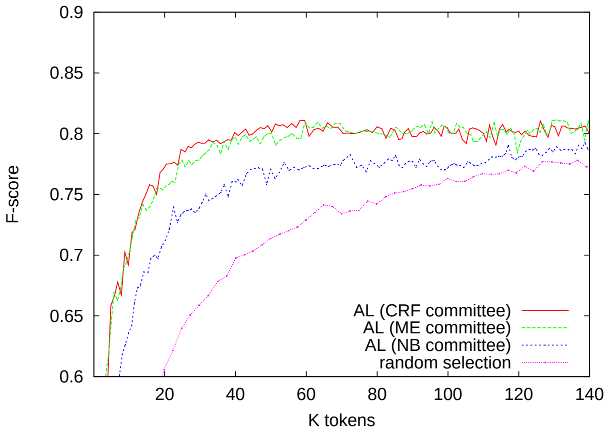


Figure 4: Algorithm Flexibility on PBvar

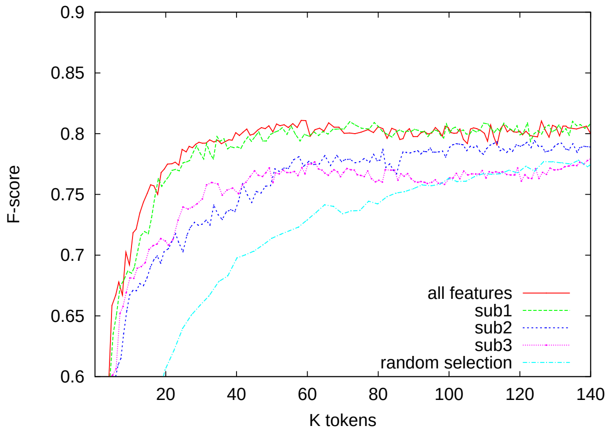


Figure 6: Feature Flexibility on PBvar

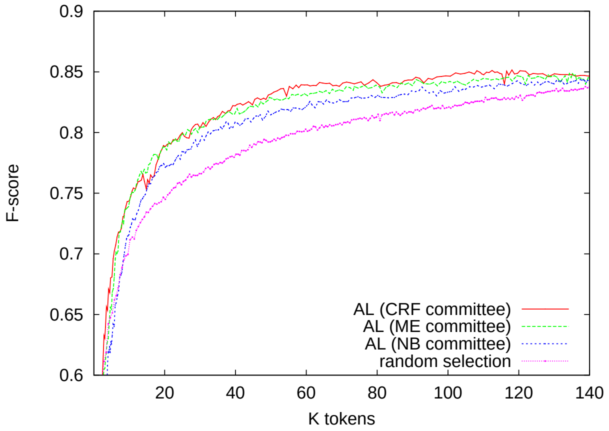


Figure 5: AlgorithmFflexibility on CoNLL

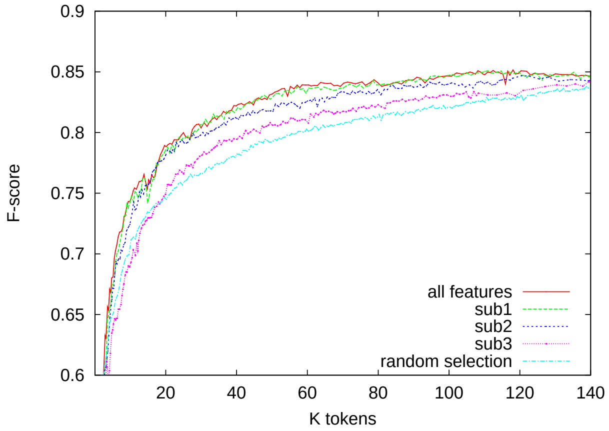


Figure 7: Feature Flexibility on ConLL

First, we analyzed the effect of different probabilistic classifiers as selectors on the resulting learning curve of the CRF tester. Figures 4 and 5 show the learning curves on our original ME committee, a CRF committee, and also a committee of Naïve Bayes (NB) classifiers. It is not surprising that self-reuse (CRF selectors and CRF tester) yields the best results. Switching from CRF selectors to ME selectors has almost no negative effect. Even with a committee of NB selectors (an ML approach which is essentially less well suited for the NER task), AL-based selection is still substantially more efficient than random selection on both corpora. This shows that our approach to use the less complex ME classifiers for the AL selection process has the positive effect of fast selection cycle times at almost no costs. This is especially interesting as the performance of an ME classifier trained in supervized manner on the complete corpus is significantly worse (several percentage points of f-measure) than a CRF. That means, even though an ME classifier is less well suited as the final classifier, it works well as a selector for CRFs.[4]

Second, we ran experiments on selectors with only some features and our CRF tester with all features (cf. Table 1). Feature subset 1 (*sub1*) contains all but the syntactic features. In the second subset (*sub2*), also morphological and lexical features are missing. The third set (*sub3*) only contains orthographical features. We ran an AL simulation for

[4] We have also conducted experiments where we varied the learning algorithms of the tester (we experimented with NB, ME, MEMM, and CRFs) – with comparable results. In a realistic scenario, however, on would rather choose a CRF as final tester over, e.g., a NB.

each feature subset with a committee of CRF selectors.[5] Figures 6 and 7 show the various learning curves. Here we see that a corpus that was produced with AL on *sub1* can easily be re-used by a tester with little more features. This is probably the most realistic scenario: the core features are kept and only a few specific features (e.g., POS, a dictionary look-up, chunk information, etc.) are added. When adding substantially more features to the tester than were available during AL time, the respective learning curves drop down towards the learning curve for random selection. But even with a selector which has only orthographical features and a tester with many more features – which is actually quite an extreme example and a rather unrealistic scenario for a real-world application – AL is more efficient than random selection. However, the limits of reusability are taking shape: on PBvar, the AL selection with *sub3* converges with the random selection curve after about 100,000 tokens.

5.3 Findings with Real AL Annotation

We currently perform AL entity mention annotations for an information extraction project in the biomedical subdomain of immunogenetics. For this purpose, we retrieved about 200,000 abstracts (≈ 2,000,000 sentences) as our document pool of unlabeled examples from PUBMED. By means of random subsampling, only about 40,000 sentences are considered in each round of AL selection. To regularly monitor classifier performance, we also perform gold standard (GS) annotations on 250 randomly chosen abstracts (≈ 2,200 sentences). In all our annotations of different entity types so far, we found AL learning curves similar to the ones reported in our simulation experiments, with classifier performance levelling off at around 75% - 85% f-score (depending on the entity type).

Our annotations also reveal that AL is especially beneficial when entity mentions are very sparse. Figure 8 shows the cumulated entity density on AL and gold standard annotations of cytokine receptors (specialized proteins for which we annotated six different entity subtypes) – very sparse entity types with less than one entity mention per PUBMED abstract on the average. As can be seen, after 2,000 sentences the entity density in our AL corpus is almost 15 times higher than in our GS corpus. Such a dense corpus may be more appropriate for classifier training than a sparse one yielded by random or sequential annotations, which may just contain lots of negative training examples. We have observed comparable effects with other entity types, too, and thus conclude that the sparser entity mentions of a specific type are in texts, the more beneficial AL-based annotation is. We report on other aspects of AL for real annotation projects in Tomanek et al. (2007).

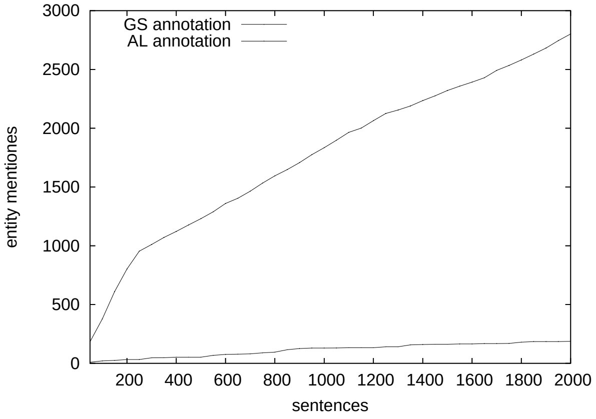


Figure 8: Cumulated Entity Density on AL and GS Annotations of Cytokine Receptors

6 Discussion and Conclusions

We have shown, for the annotation of (named) entities, that AL is well-suited to speed up annotation work under realistic conditions. In our simulations we yielded gains (in the number of tokens) up to 72%. We collected evidence that an average disagreement approaching zero may serve as an adaptive stopping criterion for AL-driven annotation and that a corpus compiled by means of QBC-based AL is to a large extent reusable by modified classifiers.

These findings stand in contrast to those supplied by Baldridge and Osborne (2004) who focused on parse selection. Their research indicates that AL on selectors with different learning algorithms and feature sets then used by the tester can easily get worse than random selection. They conclude that it might not be be advisable to employ AL in environments where the final classifier is not very stable.

Our evidence leads us to a re-assessment of AL-

[5] Here, we employed CRF instead of ME selectors to isolate the effect of feature re-usability.

based annotations. First, we employed a committee-based (QBC) while Baldridge and Osborne performed uncertainty sampling AL. Committee-based approaches calculate the uncertainty on an example in a more implicit way, i.e., by the disagreement among the committee's classifiers. With uncertainty sampling, however, the labeling uncertainty of one classifier is considered directly. In future work we will directly compare QBC and uncertainty sampling with respect to data reusability. Second, whereas Baldridge and Osborne employed AL on a scoring or ranking problem we focused on classification problems. Further research is needed to investigate whether the problem class (classification with a fixed and moderate number of classes vs. ranking large numbers of possible candidates) is responsible for limited data reusability.

On the basis of our experiments we stipulate that the proposed AL approach might be applicable with comparable results to a wider range of corpus annotation tasks, which otherwise would require substantially larger amounts of annotation efforts.

Acknowledgements

This research was funded by the EC within the BOOTStrep project (FP6-028099), and by the German Ministry of Education and Research within the StemNet project (01DS001A to 1C).

References

Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In Dekang Lin and Dekai Wu, editors, *EMNLP 2004 – Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 9–16. Barcelona, Spain, July 25-26, 2004. Association for Computational Linguistics.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT'98 – Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100. Madison, Wisconsin, USA, July 24-26, 1998. New York, NY: ACM Press.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *Journal of Artifical Intelligence Research*, 4:129–145.

Sean Engelson and Ido Dagan. 1996. Minimizing manual annotation cost in supervised training from corpora. In *ACL'96 – Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 319–326. University of California at Santa Cruz, California, U.S.A., 24-27 June 1996. San Francisco, CA: Morgan Kaufmann.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *CoNLL-2005 – Proceedings of the 9th Conference on Computational Natural Language Learning*, pages 144–151. Ann Arbor, MI, USA, June 2005. Association for Computational Linguistics.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *EMNLP/VLC-2000 – Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52. Hong Kong, China, October 7-8, 2000.. Association for Computational Linguistics.

Rebecca Hwa. 2001. On minimizing training corpus for parser acquisition. In Walter Daelemans and Rémi Zajac, editors, *CoNLL-2001 – Proceedings of the 5th Natural Language Learning Workshop*. Toulouse, France, 6-7 July 2001. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-2001 – Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Williams College, MA, USA, June 28 - July 1, 2001. San Francisco, CA: Morgan Kaufmann.

David D. Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *ICML '94: Proceedings of the 11th International Conference on Machine Learning*, pages 148–156. San Francisco, CA: Morgan Kaufmann.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *ACL'00 – Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 117–125. Hong Kong, China, 1-8 August 2000. San Francisco, CA: Morgan Kaufmann.

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In Lillian Lee and Donna Harman, editors, *EMNLP 2001 – Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 1–9. Pittsburgh, PA, USA, June 3-4, 2001. Association for Computational Linguistics.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *ICML '00: Proceedings of the 17th International Conference on Machine Learning*, pages 839–846. San Francisco, CA: Morgan Kaufmann.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *COLT'92 – Proceedings of the 5th Annual Conference on Computational Learning Theory*, pages 287–294. Pittsburgh, PA, USA, July 27-29, 1992. New York, NY: ACM Press.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *ACL'04 – Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 589–596. Barcelona, Spain, July 21-26, 2004. San Francisco, CA: Morgan Kaufmann.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *ICML '99: Proceedings of the 16th International Conference on Machine Learning*, pages 406–414. Bled, Slovenia, June 1999. San Francisco, CA: Morgan Kaufmann.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *CoNLL-2003 – Proceedings of the 7th Conference on Computational Natural Language Learning*, pages 142–147. Edmonton, Canada, 2003. Association for Computational Linguistics.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Efficient annotation with the Jena ANnotation Environment (JANE). In *Proceedings of the ACL 2007 'Linguistic Annotation Workshop – A Merger of NLPXML 2007 and FLAC 2007'*. Prague, Czech Republic, June 28-29, 2007. Association for Computational Linguistics (ACL).

Antecedent Selection Techniques for High-Recall Coreference Resolution

Yannick Versley
versley@sfs.uni-tuebingen.de
SFB 441 / Seminar für Sprachwissenschaft
Universität Tübingen

Abstract

We investigate methods to improve the recall in coreference resolution by also trying to resolve those definite descriptions where no earlier mention of the referent shares the same lexical head (coreferent bridging). The problem, which is notably harder than identifying coreference relations among mentions which have the same lexical head, has been tackled with several rather different approaches, and we attempt to provide a meaningful classification along with a quantitative comparison. Based on the different merits of the methods, we discuss possibilities to improve them and show how they can be effectively combined.

1 Introduction

Coreference resolution, the task of grouping mentions in a text that refer to the same referent in the real world, has been shown to be beneficial for a number of higher-level tasks such as information extraction (McCarthy and Lehnert, 1995), question answering (Morton, 2000) and summarisation (Steinberger et al., 2005).

While the resolution of pronominal anaphora and tracking of named entities is possible with good accuracy, the resolution of definite NPs (having a common noun as their head) is usually limited to the cases that Vieira and Poesio (2000) call direct coreference, where both coreferent mentions have the same head. The other cases, called coreferent bridging by Vieira and Poesio[1], are notably harder because the number of potential candidates is much larger when it is no longer possible to rely on surface similarity.

To overcome the limit of recall that is encountered when only relying on surface features, newer systems for coreference resolutions (Daumé III and Marcu, 2005; Ponzetto and Strube, 2006; Versley, 2006; Ng, 2007, *inter alia*) use lexical semantic information as an indication for semantic compatibility in the absence of head equality. Most current systems integrate the identification of discourse-new definites (i.e., cases like *"the sun"* or *"the man that Ben met yesterday"*, which are definite, but not anaphoric) with the antecedent selection proper, which implies that the gain obtained for new features is dependent on the feature's usefulness both in finding semantically related mentions and for the use in detecting discourse-new definites.

One goal of this paper is to provide a better understanding of these information sources by comparing proposed (and partly new) approaches for resolving coreferent bridging by separately considering the task of antecedent selection (i.e., presupposing that discourse-new markables have been identified beforehand). Although state of the art methods for modular discourse-new detection (Uryupina, 2003; Poesio et al., 2005) do not achieve near-perfect accuracy for discourse-new detection, the results we give for antecedent selection represent an upper bound on recall and precision for the full coreference task, and we think that this upper bound will be useful for

[1] Because bridging (in the sense of Clark, 1975, or Asher and Lascarides, 1998) is a much broader concept, the term 'coreferent bridging' is potentially confusing, as many cases are examples of perfectly well-behaved anaphoric definite noun phrases. Because we want to emphasise the important difference to the more easily resolved cases of same-head coreference, we will stick with 'coreferent bridging' as the only term that has been established for this in the literature.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 496–505, Prague, June 2007. ©2007 Association for Computational Linguistics

the design of features in both systems using a modular approach, such as (Poesio et al., 2005), where the decision on discourse-newness is taken beforehand, and those that integrate discourse-new classification with the actual resolution of coreferent bridging cases. In contrast to earlier investigations (Markert and Nissim, 2005; Garera and Yarowsky, 2006), we provide a more extensive overview on features and also discuss properties that influence their combinability.

Several approaches have been proposed for the treatment of coreferent bridging. Poesio et al. (1997) use WordNet, looking for a synonymy or hypernymy relation (additionally, for coordinate sisters in WordNet). The system of Cardie and Wagstaff (1999) uses the node distance in WordNet (with an upper limit of 4) as one component in the distance measure that guides their clustering algorithm. Harabagiu et al. (2001) use paths through Wordnet, using not only synonym and is-a relations, but also parts, morphological derivations, gloss texts and polysemy, which are weighted with a measure based on the relation types and number of path elements. Other approaches use large corpora to get an indication for bridging relations: Poesio et al. (1998) use a general word association metric based on common terms occuring in a fixed-width window, Gasperin and Vieira (2004) use syntactic contexts of words in a large corpus to induce a semantic similarity measure (similar to the one introduced by Lin, 1998), and then use lists of the n nouns that are (globally) most similar to a given noun. Markert and Nissim (2005) mine the World Wide Web for shallow patterns like "*China* and other *countries*", indicating an is-a relationship. Finally, Garera and Yarowsky (2006) propose an association-based approach using nouns that occur in a 2-sentence window before a definite description that has no same-head antecedent.

1.1 Lexical vs. Referential Relations

One important property of these information sources is the kind of lexical relations that they detect. The lexical relations that we expect in coreferent bridging cases are:

- instance: The antecedent is an instance of the concept denoted by the anaphor
  Corsica . . . the island

- synonymy: The antecedent and the anaphor are synonyms
  the automobile . . . the car

- hyperonymy: The anaphor is a strict generalisation of the antecedent
  the murderer . . . the man

- near-synonymy: The anaphor and antecedent are semantically related but not synonyms in the strict sense
  the CD . . . the album

Of course, not all cases of coreferent bridging realise such a lexical relation, as sometimes the anaphor takes up information introduced elsewhere than in the lexical noun phrase head (Peter was found dead in his flat . . . the deceased), or the coreference relation is forced by the discourse structure, without the items being lexically related.

As an illustrating example, in

(1) John walked towards [1 the house].

(2) a. [1 The building] was illuminated.
b. [1 The manor] was guarded by dogs.
c. [2 The door] was open.

Typical cases of coreference include cases like 1,2a (hypernym) or 1,2b (compatible but non-synonymous term). The discourse in 1,2c is an example of associative bridging between the NP *"the door"* and its antecedent to *"the house"*; it is inferred that the door must be part of the house mentioned earlier (since doors are typically part of a house), which is *not* compatible with coreferent bridging, but is also ranked highly by association measures.

While hypernym relations (as found by hypernym lookup in WordNet, or patterns indicating such relations in unannotated texts) are usually a strong indicator of coreference, they can only cover some of the cases, while the near-synonymous cases are left undiscovered. Similarity and association measures can help for the cases of near-synonymy. However, while similarity measures (such as WordNet distance or Lin's similarity metric) only detect cases of semantic similarity, association measures (such as the ones used by Poesio et al., or by Garera and Yarowsky) also find cases of associative bridg-

Lin98	RFF	TheY	TheY:G^2	PL03
Land ***(country/state/land)***				
Staat *state*	Staat *state*	Kemalismus *Kemalism*	Regierung *government*	Kontinent *continent*
Stadt *city*	Stadt *city*	Bauernfamilie *agricultural family*	Präsident *president*	Region *region*
Region *region*	Landesregierung *country government*	Bankgesellschaft *banking corporation*	Dollar *dollar*	Stadt *city*
Bundesrepublik *federal republic*	Bundesregierung *federal government*	Baht *Baht*	Albanien *Albania*	Staat *state*
Republik *republic*	Gewerkschaft *trade union*	Gasag *(a gas company)*	Hauptstadt *capital*	Bundesland *state*
Medikament ***(medical drug)***				
Arzneimittel *pharmaceutical*	Pille *pill*	RU *(a drug***)*	Patient *patient*	Arzneimittel *pharmaceutical*
Präparat *preparation*	Droge *drug (non-medical)*	Abtreibungspille *abortion pill*	Arzt *doctor*	Lebensmittel *foodstuff*
Pille *pill*	Präparat *preparation*	Viagra *Viagra*	Pille *pill*	Präparat *preparation*
Hormon *hormone*	Pestizid *pesticide*	Pharmakonzern *pharmaceutical company*	Behandlung *treatment*	Behandlung *treatment*
Lebensmittel *foodstuff*	Lebensmittel *foodstuff*	Präparat *preparation*	Abtreibungspille *abortion pill*	Arznei *drug*

highest ranked words, with very rare words removed

*: RU 486, an abortifacient drug

Lin98: Lin's distributional similarity measure (Lin, 1998)

RFF: Geffet and Dagan's *Relative Feature Focus* measure (Geffet and Dagan, 2004)

TheY: association measure introduced by Garera and Yarowsky (2006)

TheY:G^2: similar method using a log-likelihood-based statistic (see Dunning 1993)
this statistic has a preference for higher-frequency terms

PL03: semantic space association measure proposed by Padó and Lapata (2003)

Table 1: Similarity and association measures: most similar items

ing like 1a,b; the result of this can be seen in table (2): while the similarity measures (Lin98, RFF) list substitutable terms (which behave like synonyms in many contexts), the association measures (Garera and Yarowsky's TheY measure, Padó and Lapata's association measure) also find non-compatible associations such as *country–capital* or *drug–treatment*, which is why they are commonly called *relation-free*. For the purpose of coreference resolution, however we do *not* want to resolve *"the door"* to the antecedent *"the house"* as the two descriptions do not corefer, and it may be useful to filter out non-similar associations.

1.2 Information Sources

Different resources may be differently suited for the recognition of the various relations. Generally, it would be expected that using a wordnet is the best solution if we are interested in an isa-like relation between two words. On the other hand, wordnets usually have limited coverage both in terms of lexical items and in terms of relations encoded (as their construction is necessarily labor-intensive), and – as Markert and Nissim remark – they do not (and arguably should not) contain context-dependent relations that do not hold generally but only in some rather specific context, for example *steel* being anaphorically described as a *commodity* in a financial text. Context-dependent relations, Markert and Nissim argue, can be found using shallow patterns (for example, *steel and other commodities*), since a use in such a context would mean that the idiosyncratic conceptual relation holds in that context. Wordnets also have usually have poor (or non-existant) coverage of named entities, which are especially relevant for instance relations; this kind of instance relations can often be found in large text corpora. The high-precision patterns that Markert and Nissim use only occur infrequently, but the approach using shallow patterns allows to perform the search of the World Wide Web, which somewhat alleviates the sparse data problem.

While some near-synonyms can be found by looking at the distance in a wordnet, they may be far apart from each other because of ontological modeling decisions, or lexical items not covered by the wordnet. Similarity and association measures can provide greater coverage for these near-synonym relations.

The measures both of Lin (1998) and of Padó and Lapata (2003, 2007) are distributional methods; for each word, they create a distribution of the contexts they occur in, and similarity between two words is calculated as the similarity of these distributions.[2] The difference in these two methods is the representation of the contexts. While Lin uses contexts that are expected to determine semantic preferences (like being in the direct object position of one verb), Padó and Lapata only use the co-occuring words, weighted by syntax-based distance. For example, in

(3) Peter $\overset{subj}{\rightarrow}$ likes $\overset{dobj}{\leftarrow}$ ice-cream.

Lin's approach would yield $\uparrow subj{:}\texttt{like}$ for `Peter` and $\uparrow dobj{:}\texttt{like}$ for `ice-cream`, while Padó and Lapata's approach would yield the contexts `like` (with a weight of 1.0) and `ice-cream` (with a weight of 0.5) for `Peter`. As a consequence, Padó and Lapata's measure is more robust against data sparseness but also finds related non-similar terms (which are ultimately unwanted for coreference resolution). Padó and Lapata show their dependency-based measure to perform better in a word sense disambiguation task than the measure of Lund et al. (1995), on which Poesio et al. (1998) based their experiments and which is based on the surface distance of words.

We also reimplemented the approach of Garera and Yarowsky (2006), who extract potential anaphor-antecedent pairs from unlabeled texts and rank these potentially related pairs by the mutual information statistic. As an example, in a text like

(4) Peter likes ice-cream.
The boy devours tons of it.

we would extract the pairs $\langle \texttt{boy}, \texttt{(person)} \rangle$ and $\langle \texttt{boy}, \texttt{ice-cream} \rangle$, in the hope that the former pair occurs comparatively more often and gets a higher mutual information value.

2 Experiments on Antecedent Selection

In a setting similar to Markert and Nissim (2005), we evaluate the precision (proportion of correct cases in the resolved cases) and recall (correct cases to all cases) for the resolution of discourse-old definite noun phrases. Before trying to resolve coreferent bridging cases, we look for compatible antecedent candidates with the same lexical head and resolve to the nearest such candidate if there is one.

For our experiments, we used the first 125 articles of the coreferentially annotated TüBa-D/Z corpus of written newspaper text (Hinrichs et al., 2005), totalling 2239 sentences with 633 discourse-old definite descriptions, and the latest release of GermaNet (Kunze and Lemnitzer, 2002), which is the German-language part of EuroWordNet.

Unlike Markert and Nissim, we did not limit the evaluation to discourse-old noun phrases where an antecedent is in the 4 preceding sentences, but also included cases where the antecedent is further away. As a real coreference resolution system would have to either resolve them correctly or leave them unresolved, we feel that this is less unrealistic and thus preferable even when it gives less optimistic evaluation results. Because overall precision is a mixture of the precision of the same-head resolver and the precision of the resolution for coreferent bridging, which is lower than that for same-head cases, we forcibly get less precision if we resolve more coreferent bridging cases. As it is always possible to improve overall precision by resolving fewer cases of coreferent bridging, we separately mention the precision for coreferent bridging cases alone (i.e., number of correct coreferent bridging cases by all resolved coreferent bridging cases), which we deem more informative.

In our evaluation, we included hypernymy search and a simple edge-based distance based on GermaNet, as well as a baseline using semantic classes (automatically determined by a combination of simple named entity classification and GermaNet subsumption), as well as an evolved version of Markert

[2] Both measures use a weighted Jaccard metric on mutual information vectors to calculate the similarity. See Weeds and Weir (2005) for an overview of other measures.

	Prec	Recl	$F_{\beta=1}$	Prec.NSH
same-head	0.87	0.50	0.63	—
nearest$^{(1)}$ (only number check)	0.57	0.55	0.56	0.12
semantic class+gender check$^{(1)}$	0.68	0.61	0.64	0.35
semantic class+gender check$^{(2)}$	0.67	0.62	0.65	0.36
GermaNet, hypernymy lookup	**0.83**	0.58	**0.68**	**0.67**
GermaNet, node distance$^{(1)}$	0.71	0.61	0.65	0.39
single pattern: "Y wie X"$^{(1)}$	0.83	0.54	0.66	0.55
TheY$^{(1)}$ (only number checking)	0.66	0.59	0.62	0.29
TheY$^{(2)}$ (only number checking)	0.66	0.60	0.63	0.31
Lin$^{(1)}$ (only number checking)	0.66	0.60	0.63	0.30
Lin$^{(2)}$ (only number checking)	0.69	0.64	0.66	0.39
PL03$^{(1)}$ (only number checking)	0.68	0.63	0.65	0.38
PL03$^{(2)}$ (only number checking)	0.70	**0.64**	0.65	0.42
15-most-similar$^{(1)}$	0.82	0.54	0.65	0.50
100-most-similar$^{(2,3)}$	0.73	0.60	0.66	0.42

Prec.NSH: precision for coreferent bridging cases

$^{(1)}$: consider candidates in the 4 preceding sentences

$^{(2)}$: consider candidates in the 16 preceding sentences

$^{(3)}$: also try candidates such that the anaphor is in the antecedent's similarity list

Table 2: Baseline results

and Nissim's approach, which is presented in (Versley, 2007). For the methods based on similarity and association measures, we implemented a simple ranking by the respective similarity or relatedness value. Additionally, we included an approach due to Gasperin and Vieira (2004), who tackle the problem of similarity by using lists of most similar words to a certain word, based on a similarity measure closely related to Lin's. They allow resolution if either (i) the candidate is among the words most similar to the anaphor, (ii) the anaphor is among the words most similar to the candidate, (iii) the similarity lists of anaphor and candidate share a common item. We tried out several variations in the length of the similar words list (Gasperin and Vieira used 15, we also tried lists with 25, 50 and 100 items). The third possibility that Gasperin and Vieira mention (a common item in the similarity lists of both anaphor and antecedent) resolves some correct cases, but leads to a much larger number of false positives, which is why we did not include it in our evaluation.

To induce the similarity and association measures presented earlier, we used texts from the German newspaper *die tageszeitung*, comprising about 11M sentences. For the extraction of anaphor-antecedent candidates, we used a chunked version of the corpus (Müller and Ule, 2002). The identification of grammatical relations, was carried out on a subset of all sentences (those with length ≤ 30), with an unlexicalised PCFG parser and subsequent extraction of dependency relations (Versley, 2005). For the last approach, where dependency relations were needed but labeling accuracy was not as important, we used a deterministic shift-reduce parser that Foth and Menzel (2006) used as input source in hybrid dependency parsing.[3]

For all three approaches, we lemmatised the words by using a combination of SMOR (Schmid et al., 2004), a derivational finite-state morphology for German, and lexical information derived from the lexicon of a German dependency parser (Foth and Menzel, 2006). We mitigated the problem of vocabulary growth in the lexicon, due to German synthetic compounds, by using a frequency-sensitive unsupervised compound splitting technique, and (for semantic similarity) normalised common person and location names to '(person)' and '(location)', respectively.

Same-head resolution (including a check for modifier compatibility) allows to correctly resolve 49.8% of all cases, with a precision of 86.5%. The most simple approach for coreferent bridging, just resolving coreferent bridging cases to the nearest possible antecedent (only checking for number agreement), yields very poor precision (12% for the coreferent bridging cases), and as a result, the recall gain is very limited. If we use semantic classes (based on both GermaNet and a simple classification for named entities) to constrain the candidates and then use the nearest number- and gender-compatible antecedent[4], we get a much better precision (35% for coreferent bridging cases), and a much better recall of 61.1%. Hyponymy lookup in GermaNet, without a limit on sentence distance, achieves a recall of 57.5% (with a precision of 67% for the resolved coreferent bridging cases), whereas using the best single pattern (Y wie X, which corresponds to

[3] Arguably, it would have been more convenient to use a single parser for all three approaches, but differing tradeoffs between speed on one hand and accuracy for relevant information and/or fitness of representation on the other hand made the respective parser or chunker a compelling choice.

[4] In German, grammatical gender is not as predictive as in English as it does not reproduce ontological distinctions. For persons, grammatical and natural gender almost always coincide, and we check gender equality iff the anaphor is a person.

the English Ys such as X), with a distance limit of 4 sentences[5], on the Web only improves the recall to 54.3% (with a lower precision of 55% for coreferent bridging cases). This is in contrast to the results of Markert and Nissim, who found that Web pattern search performs better than wordnet lookup; see (Versley, 2007) for a discussion. Ranking all candidates that are within a distance of 4 hyper-/hyponymy edges in GermaNet by their edge distance, we get a relatively good recall of 60.5%, but the precision (for the coreferent bridging cases) is only at 39%, which is quite poor in comparison.

The results for Garera and Yarowsky's TheY algorithm are quite disconcerting – recall and the precision on coreferent bridging cases are lower than the respective baseline using (wordnet-based) semantic class information or Padó and Lapata's association measure. The technique based on Lin's similarity measure does outperform the baseline, but still suffers from bad precision, along with Padó and Lapata's association measure. In other words, the similarity and association measures seem to be too noisy to be used directly for ranking antecedents. The approach of Gasperin and Vieira performs comparably to the approach using Web-based pattern search (although the precision is poorer than for the best-performing pattern for German, "X wie Y" – X such as Y, it is comparable to that of other patterns).

2.1 Improving Distributional Similarity?

While it would be naïve to think that the methods purely based on statistical similarity measures could reach the accuracy that can be achieved with a hand-constructed lexicalised ontology, it would of course be nice if we could improve the quality of the semantic similarity measure used in ranking and the most-similar-word lists.

Geffet and Dagan (2004) propose an approach to improve the quality of the feature vectors used in distributional similarity measures: instead of weighting features using the mutual information value between the word and the feature, they propose to use a measure they call *Relative Feature Focus*: the sum of the similarities to the (globally) most similar words that share this feature.

By replacing mutual information values with RFF values in Lin's association measure, Geffet and Dagan were able to significantly improve the proportion of substitutable words in the list of the most similar words. In our experiments, however, using the RFF-based similarity measure did not improve the similarity-list-based resolution or the simple ranking, to the contrary, both recall and precision are less than for the Weighted Jaccard measure that we used originally.[6]

We attribute this to two factors: Firstly, Geffet and Dagan's evaluation emphasises the precision in terms of *types*, whereas the use in resolving coreferent bridging does not punish unrelated rare words being ranked high – since these are rare, the likelihood that they occur together, changing a resolution decision, is quite low, whereas rare related words that are ranked high can allow a correct resolution. Secondly, Geffet and Dagan focus on high-frequency words, which makes sense in the context of ontology learning, but the applicability for tasks like coreference resolution (directly or in the approach of Gasperin and Vieira) also depends on a sensible treatment of lower-frequency words.

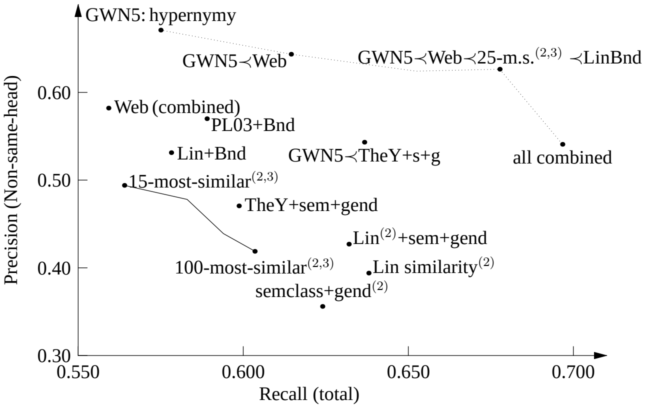
Using the framework of Weeds et al. (2004), we found that the bias of lower frequency words for preferring high-frequency neighbours was higher for RFF (0.58 against 0.35 for Lin's measure). Weeds and Weir (2005) discuss the influence of bias towards high- or low-frequency items for different tasks (correlation with WordNet-derived neighbour sets and pseudoword disambiguation), and it would not be surprising if the different high-frequency bias were leading to different results.

2.2 Combining Information Sources

The information sources that we presented earlier and the corpus-based methods based on similarity or association measures draw from different kinds of evidence and thus should be rather complementary. To put it another way, it should be possible to get the best from all methods, achieving the recall of the high-recall methods (like using semantic class in-

[5] There is a degradation in precision for the pattern-based approach, but not for the GermaNet-based approach, which is why we do not use a distance limit for the GermaNet-based approach.

[6] Simple ranking with RFF gives a precision of 33% for coreferent bridging cases, against 39% for Lin's original measure; for an approach based on similarity lists, we get 39% against 44%.



	Prec	Recl	$F_{\beta=1}$	Prec.NSH
sem. class+gender checking	0.68	0.61	0.64	0.35
GermaNet, hypernymy lookup	0.83	0.57	0.68	0.67
GermaNet $\prec$ "Y wie X"	0.81	0.60	0.69	0.63
GermaNet $\prec$ all patterns	0.81	0.61	0.70	0.64
TheY$^{(2)}$+semclass+gender	0.76	0.60	0.67	0.47
TheY+sem+gend+Bnd	0.78	0.59	0.67	0.50
Lin$^{(2)}$+semclass+gender	0.71	0.63	0.67	0.43
Lin+sem+gend+Bnd	0.80	0.58	0.67	0.53
PL03$^{(2)}$+semclass+gender	0.72	0.64	0.68	0.45
PL03+sem+gend+Bnd	0.80	0.59	0.68	0.57
GermaNet $\prec$ all patterns	0.81	0.62	0.70	0.64
$\prec$ 25-most-similar$^{(2,3)}$	0.79	0.65	0.72	0.62
$\prec$ LinBnd	0.79	0.68	**0.73**	0.63
$\prec$ Lin $\prec$ TheY+sem+gend	0.74	**0.70**	0.72	0.54

$^{(2)}$: consider candidates in the 16 preceding sentences

$^{(3)}$: also try candidates such that the anaphor is in the antecedent's similarity list

Table 3: Combination-based approaches

formation, or similarity and association measures), with a precision closer to the most precise method using GermaNet. In the case of web-based patterns, Versley (2007) combines several pattern searches on the web and uses the combined positive and negative evidence to compute a composite score – with a suitably chosen cutoff, it outperforms all single patterns both in terms of precision and recall. First resolving via hyponymy in GermaNet and then using the pattern-combination approach outperforms the semantic class-based baseline in terms of recall and is reasonably close to the GermaNet-based approach in terms of precision (i.e., much better than the approach based only on the semantic class).

As a first step to improve the precision of the corpus-based approaches, we added filtering based on automatically assigned semantic classes (persons, organisations, events, other countable objects, and everything else). Very surprisingly, Garera and Yarowsky's TheY approach, despite starting out at a lower precision (31%, against 39% for Lin and 42% for PL03), profits much more from the semantic filter and reaches the best precision (47%), whereas Lin's semantic similarity measure profits the least.

Since limiting the distance to the 4 previous sentences had quite a devastating effect for the approach based on Lin's similarity measure (which achieves 39% precision when all the candidates are available and 30% precision if it choses the most semantically similar out of the candidates that are in the last 4 sentences), we also wanted to try and apply the distance-based filtering after finding semantically related candidates.

The approach we tried was as follows: we rank all candidates using the similarity function, and keep only the 3 top-rated candidates. From these 3 top-rated candidates, we keep only those within the last 4 sentences. Without filtering by semantic class, this improves the precision to 41% (from 30% for limiting the distance beforehand, or 39% without limiting the distance). Adding filtering based on semantic classes to this (only keeping those from the 3 top-rated candidates which have a compatible semantic class and are within the last 4 sentences), we get a much better precision of 53%, with a recall that can still be seen as good (57.8%). In comparison with the similarity-list-based approach, we get a much better precision than we would get for methods with comparable recall (the version with the 100 most similar items has 44% precision, the version with 50 most similar items and matching both ways has 46% precision).

Applying this distance-bounding method to Garera and Yarowsky's association measure still leads to an improvement over the case with only semantic and gender checking, but the improvement (from 47% to 50%) is not as large as with the semantic similarity measure or Padó and Lapata's association measure (from 45% to 57%).

For the final system, we back off from the most precise information sources to the less precise. Starting with the combination of GermaNet and pattern-based search on the World Wide Web, we begin by adding the distance-bounded semantic similarity-based resolver (LinBnd) and resolution based on the list of 25 most similar words (following the approach of Gasperin and Vieira 2004). This results in visibly improved recall (from 62% to 68%), while the precision for coreferent bridging cases does not suffer much. Adding resolution based on Lin's semantic similarity measure and Garera and Yarowsky's TheY value leads to a further improvement in recall to 69.7%, but also leads to a larger loss in precision.

3 Conclusion

In this paper, we compared several approaches to resolve cases of coreferent bridging in open-domain newspaper text. While none of the information sources can match the precision of the hypernymy information encoded in GermaNet, or that of using a combination of high-precision patterns with the World Wide Web as a very large corpus, it is possible to achieve a considerable improvement in terms of recall without sacrificing too much precision by combining these methods.

Very interestingly, the distributional methods based on intra-sentence relations (Lin, 1998; Padó and Lapata, 2003) outperformed Garera and Yarowsky's (2006) association measure when used for ranking, which may due to sparse data problems or simply too much noise for the latter. For the association measures, the fact that they are relation-free also means that they can profit from added semantic filtering.

The novel distance-bounded semantic similarity method (where we use the most similar words in the previous discourse together with a semantic class-based filter and a distance limit) comes near the precision of using surface patterns, and offers better accuracy than Gasperin and Vieira's method of using the globally most similar words.

By combining existing higher-precision information sources such as hypernym search in GermaNet and the Web-based approach presented in (Versley, 2007) together with similarity- and association-based resolution, it is possible to get a large improvement in recall even compared to the combined GermaNet+Web approach or an approach combining GermaNet with a semantically filtered version of Garera and Yarowsky's TheY approach.

In independent research, Goecke et al. (2006) combined the original LSA-based method of Lund et al. (1995) with wordnet relations and pattern search on a fixed-size corpus.[7] However, they evaluate only on a small subset of discourse-old definite descriptions (those where a wordnet-compatible semantic relation was identified and which were reasonably close to their antecedent), and they did not distinguish coreferent from associative bridging antecedents. Although the different evaluation method disallows a meaningful comparison, we think that the more evolved information sources we use (Padó and Lapata's association measure instead of Lund et al's, combined pattern search on the World Wide Web instead of search for patterns in a fixed-size corpus), as well as the additional information based on semantic similarity, lead to superior results when evaluated in a comparable task.

3.1 Ongoing and Future Work

Both the distributional similarity statistics and the association measure can profit from more training data, something which is bound by availability of similar text (Gasperin et al., 2004 point out that using texts from a different genre strongly limits the usefulness of the learned semantic similarity measure), and by processing costs (which are more serious for distributional similarity measures than for non-grammar-related association measures, as the former necessitate parsed input).

Based on existing results for named entity coreference, a hypothetical coreference resolver combining our information sources with a perfect detector for discourse-new mentions would be able to achieve a precision of 88% and a recall of 83% considering all full noun phrases (i.e., including names, but not pronouns). This is both much higher than state-of-the art results for the same data set (Versley, 2006, gets 62% precision and 70% recall), but such accuracy may be very difficult to achieve in practice, as perfect (or even near-perfect) discourse-new detection does not seem to achievable in the near future. Preliminary experiments show that the integration of pattern-based information leads to an increase in recall of 0.6% for the whole system (or 46% more coreferent bridging cases), but the integration of distributional similarity (loosely based on the approach by Gasperin and Vieira) does not lead

[7] Thanks to Tonio Wandmacher for pointing this out to me at GLDV'07.

to a noticeable improvement over GermaNet alone; in isolation, the distributional similarity information did improve the recall, albeit less than information from GermaNet did.

The fact that only a small fraction of the achievable recall gain is currently attained seems to suggest that better identification of discourse-old mentions could potentially lead to larger improvements. It also seems that firstly, it makes more sense to combine information sources that cover different relations (e.g. GermaNet for hypernymy and synonymy and the pattern-based approach for instance relations) than those that yield independent evidence for the same relation(s), as GermaNet and the Gasperin and Vieira approach do for (near-)synonymy; and secondly, that good precision is especially important in the context of integrating antecedent selection and discourse-new identification, which means that the finer view that we get using antecedent selection experiments (compared to direct use in a coreference resolver) is indeed helpful.

Acknowledgements I am very grateful to Sabine Schulte im Walde, Piklu Gupta and Sandra Kübler for useful criticism of an earlier version, and to Simone Ponzetto and Michael Strube for feedback on a talk related to this paper. The research reported in this paper was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of Collaborative Research Centre (Sonderforschungsbereich) 441 “Linguistic Data Structures”.

References

Asher, N. and Lascarides, A. (1998). Bridging. *Journal of Semantics*, 15(1):83–113.

Cardie, C. and Wagstaff, K. (1999). Noun phrase coreference as clustering. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC 1999)*, pages 82–89.

Clark, H. H. (1975). Bridging. In Schank, R. C. and Nash-Webber, B. L., editors, *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 169–174, Cambridge, MA. Association for Computing Machinery.

Daumé III, H. and Marcu, D. (2005). A large-scale exploration of effective global features for a joint entity detection and tracking model. In *HLT/EMNLP'05*, pages 97–104.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Foth, K. and Menzel, W. (2006). Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *ACL 2006*.

Garera, N. and Yarowsky, D. (2006). Resolving and generating definite anaphora by modeling hypernymy using unlabeled corpora. In *CoNLL 2006*.

Gasperin, C., Salmon-Alt, S., and Vieira, R. (2004). How useful are similarity word lists for indirect anaphora resolution? In *Proc. DAARC 2004*.

Gasperin, C. and Vieira, R. (2004). Using word similarity lists for resolving indirect anaphora. In *ACL'04 workshop on reference resolution and its applications*.

Geffet, M. and Dagan, I. (2004). Feature vector quality and distributional similarity. In *CoLing 2004*.

Goecke, D., Stührenberg, M., and Wandmacher, T. (2006). Extraction and representation of semantic relations for resolving definite descriptions. In *Workshop on Ontologies in Text Technology (OTT 2006)*. extended abstract.

Harabagiu, S., Bunescu, R., and Maiorano, S. (2001). Text and knowledge mining for coreference resolution. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association of Computational Linguistics (NAACL-2001)*.

Hinrichs, E., Kübler, S., and Naumann, K. (2005). A unified representation for morphological, syntactic, semantic and referential annotations. In *ACL Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, Ann Arbor.

Kunze, C. and Lemnitzer, L. (2002). Germanet – representation, visualization, application. In *Proceedings of LREC 2002*.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proc. CoLing/ACL 1998*.

Lund, K., Atchley, R. A., and Burgess, C. (1995). Semantic and associative priming in high-dimensional semantic space. In *Proc. of the 17th Annual Conference of the Cognitive Science Society*, pages 660–665.

Markert, K. and Nissim, M. (2005). Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31(3):367–402.

McCarthy, J. F. and Lehnert, W. G. (1995). Using decision trees for coreference resolution. In *IJCAI 1995*, pages 1050–1055.

Morton, T. S. (2000). Coreference for NLP applications. In *ACL-2000*.

Müller, F. H. and Ule, T. (2002). Annotating topological fields and chunks – and revising POS tags at the same time. In *Proceedings of the Nineteenth International Conference on Computational Linguistics (COLING 2002)*.

Ng, V. (2007). Shallow semantics for coreference resolution. In *IJCAI 2007*, pages 1689–1694.

Padó, S. and Lapata, M. (2003). Constructing semantic space models from parsed corpora. In *Proceedings of ACL 2003*.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, to appear.

Poesio, M., Alexandrov-Kabadjov, M., Vieira, R., Goulart, R., and Uryupina, O. (2005). Does discourse-new detection help definite description resolution? In *Proceedings of the 6th International Workshop on Computational Semantics (IWCS-6)*.

Poesio, M., Schulte im Walde, S., and Brew, C. (1998). Lexical clustering and definite description interpretation. In *AAAI Spring Symposium on Learning for Discourse*.

Poesio, M., Vieira, R., and Teufel, S. (1997). Resolving bridging descriptions in unrestricted text. In *ACL-97 Workshop on Operational Factors in Practical, Robust, Anaphora Resolution For Unrestricted Texts*.

Ponzetto, S. P. and Strube, M. (2006). Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *HLT-NAACL 2006*.

Schmid, H., Fitschen, A., and Heid, U. (2004). SMOR: A german computational morphology covering derivation, composition and inflection. In *Proceedings of LREC 2004*.

Steinberger, J., Kabadjov, M., Poesio, M., and Sanchez-Graillet, O. (2005). Improving LSA-based summarization with anaphora resolution. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 1–8.

Uryupina, O. (2003). High-precision identification of discourse new and unique noun phrases. In *Proceedings of the ACL Student Workshop*.

Versley, Y. (2005). Parser evaluation across text types. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*.

Versley, Y. (2006). A constraint-based approach to noun phrase coreference resolution in German newspaper text. In *Konferenz zur Verarbeitung Natürlicher Sprache (KONVENS 2006)*.

Versley, Y. (2007). Using the Web to resolve coreferent bridging in German newspaper text. In *Proceedings of GLDV-Frühjahrstagung 2007*, Tübingen. Narr.

Vieira, R. and Poesio, M. (2000). An empirically based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

Weeds, J. and Weir, D. (2005). Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics*, 31(4):439–475.

Weeds, J., Weir, D., and McCarthy, D. (2004). Characterizing measures of lexical distributional similarity. In *CoLing 2004*.

Methods to integrate a language model with semantic information for a word prediction component

Tonio Wandmacher
Laboratoire d'Informatique (LI)
Université François Rabelais de Tours
3 place Jean-Jaurès, 41000 Blois, France
tonio.wandmacher@
univ-tours.fr

Jean-Yves Antoine
Laboratoire d'Informatique (LI)
Université François Rabelais de Tours
3 place Jean-Jaurès, 41000 Blois, France
jean-yves.antoine@
univ-tours.fr

Abstract

Most current word prediction systems make use of n-gram language models (LM) to estimate the probability of the following word in a phrase. In the past years there have been many attempts to enrich such language models with further syntactic or semantic information. We want to explore the predictive powers of Latent Semantic Analysis (LSA), a method that has been shown to provide reliable information on long-distance semantic dependencies between words in a context. We present and evaluate here several methods that integrate LSA-based information with a standard language model: a *semantic cache*, *partial reranking*, and different forms of interpolation. We found that all methods show significant improvements, compared to the 4-gram baseline, and most of them to a simple cache model as well.

1 Introduction: NLP for AAC systems

Augmented and Alternative Communication (AAC) is a field of research which concerns natural language processing as well as human-machine interaction, and which aims at restoring the communicative abilities of disabled people with severe speech and motion impairments. These people can be for instance cerebrally and physically handicapped persons or they suffer from a locked-in syndrome due to a cerebral apoplexy. Whatever the disease or impairment considered, oral communication is impossible for these persons who have in addition serious difficulties to control physically their environment. In particular, they are not able to use standard input devices of a computer. Most of the time, they can only handle a single switch device. As a result, communicating with an AAC system consists of typing messages by means of a virtual table of symbols (words, letters or icons) where the user successively selects the desired items.

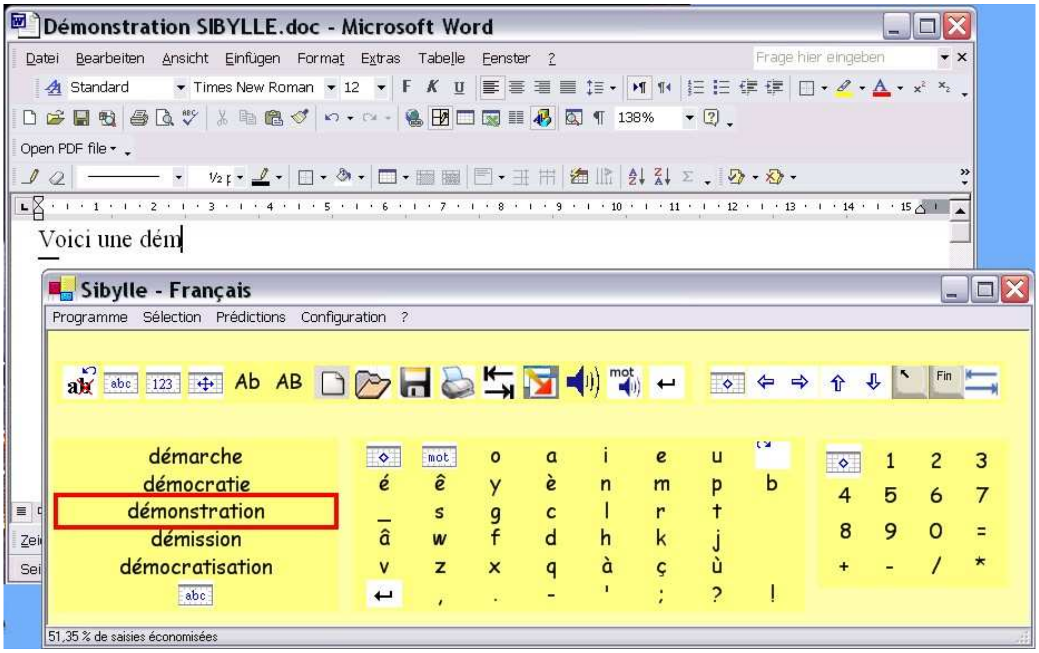
Basically, an AAC system, such as *FASTY* (Trost et al. 2005) or *SIBYLLE* (Schadle et al, 2004), consists of four components. At first, one finds a physical input interface connected to the computer. This device is adapted to the motion capacities of the user. When the latter must be restricted to a single switch (eye glimpse or breath detector, for instance), the control of the environment is reduced to a mere *Yes/No* command.

Secondly, a virtual keyboard is displayed on screen. It allows the user to select successively the symbols that compose the intended message. In *SIBYLLE*, key selection is achieved by pointing letters through a linear scan procedure: a cursor successively highlights each key of the keyboard.

The last two components are a text editor (to write e-mails or other documents) and a speech synthesis module, which is used in case of spoken communication. The latest version of *SIBYLLE* works for French and German, and it is usable with any *Windows*™ application (text editor, web browser, mailer...), which means that the use of a specific editor is no longer necessary.

The main weakness of AAC systems results from the slowness of message composition. On average, disabled people cannot type more than 1 to 5 words per minute; moreover, this task is very tiring. The use of NLP techniques to improve AAC systems is therefore of first importance.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 506–513, Prague, June 2007. ©2007 Association for Computational Linguistics


Démonstration SIBYLLE.doc - Microsoft Word
Datei Bearbeiten Ansicht Einfügen Format Extras Tabelle Fenster ?
Frage hier eingeben
Standard
Times New Roman
12
138%
Open PDF file
Sibylle - Français
Programme Sélection Prédictions Configuration ?
démarche
démocratie
démonstration
démission
démocratisation
abc
mot
Fin

Voici une dém

51,35 % de saisies économisées

Figure 1: User interface of the *SIBYLLE* AAC system

Two complementary approaches are possible to speed up communication. The first one aims at minimizing the duration of each item selection. Considering a linear scan procedure, one could for instance dynamically reorganize the keyboard in order to present the most probable symbols at first. The second strategy tries to minimize the number of keystrokes to be made. Here, the system tries to predict the words which are likely to occur just after those already typed. The predicted word is then either directly displayed after the end of the inserted text (a method referred to as "word completion", cf. Boissière and Dours, 1996), or a list of N-best (typically 3 to 7) predictions is provided on the virtual keyboard. When one of these predictions corresponds to the intended word, it can be selected by the user. As can be seen in figure 1, the interface of the *SIBYLLE* system presents such a list of most probable words to the user.

Several approaches can be used to carry out word prediction. Most of the commercial AAC systems make only use of a simple lexicon: in this approach, the context is not considered.

On the other hand, stochastic language models can provide a list of word suggestions, depending on the *n-1* (typically *n* = 3 or 4) last inserted words. It is obvious that such a model cannot take into account long-distance dependencies. There have been attempts to integrate part-of-speech information (Fazly and Hirst, 2003) or more complex syntactic models (Schadle et al, 2004) to achieve a better prediction. In this paper, we will nevertheless limit our study to a standard 4-gram model as a baseline to make our results comparable. Our main aim is here to investigate the use of long-distance semantic dependencies to dynamically adapt the prediction to the current semantic context of communication. Similar work has been done by Li and Hirst (2005) and Matiasek and Baroni (2003), who exploit *Pointwise Mutual Information* (PMI; Church and Hanks, 1989). Trnka et al. (2005) dynamically interpolate a high number of topic-oriented models in order to adapt their predictions to the current topic of the text or conversation.

Classically, word predictors are evaluated by an objective metric called *Keystroke Saving Rate* (*ksr*):

$$ksr_n = \left(1 - \frac{k_p}{k_a}\right) \cdot 100 \qquad (1)$$

with k_p, k_a being the number of keystrokes needed on the input device when typing a message with (k_p) and without prediction (k_a = number of characters in the text that has been entered, *n* = length of the prediction list, usually *n* = 5). As

Trost et al. (2005) and Trnka et al. (2005), we assume that one additional keystroke is required for the selection of a word from the list and that a space is automatically inserted afterwards. Note also that words, which have already occurred in the list, will not reappear after the next character has been inserted.

The perplexity measure, which is frequently used to assess statistical language models, proved to be less accurate in this context. We still present perplexities as well in order to provide comparative results.

2 Language modeling and semantics

2.1 Statistical Language Models

For about 10 to 15 years statistical language modeling has had a remarkable success in various NLP domains, for instance in speech recognition, machine translation, Part-of-Speech tagging, but also in word prediction systems. N-gram based language models (LM) estimate the probability of occurrence for a word, given a string of *n*-1 preceding words. However, computers have only recently become powerful enough to estimate probabilities on a reasonable amount of training data. Moreover, the larger *n* gets, the more important the problem of combinatorial explosion for the probability estimation becomes. A reasonable trade-off between performance and number of estimated events seems therefore to be an *n* of 3 to 5, including sophisticated techniques in order to estimate the probability of unseen events (smoothing methods).

Whereas n-gram-like language models are already performing rather well in many applications, their capacities are also very limited in that they cannot exploit any deeper linguistic structure. Long-distance syntactic relationships are neglected as well as semantic or thematic constraints.

In the past 15 years many attempts have been made to enrich language models with more complex syntactic and semantic models, with varying success (cf. (Rosenfeld, 1996), (Goodman, 2002) or in a word prediction task: (Fazly and Hirst, 2003), (Schadle, 2004), (Li and Hirst, 2005)). We want to explore here an approach based on *Latent Semantic Analysis* (Deerwester et al, 1990).

2.2 Latent Semantic Analysis

Several works have suggested the use of *Latent Semantic Analysis* (LSA) in order to integrate semantic similarity to a language model (cf. Bellegarda, 1997; Coccaro and Jurafsky, 1998). LSA models semantic similarity based on co-occurrence distributions of words, and it has shown to be helpful in a variety of NLP tasks, but also in the domain of cognitive modeling (Landauer et al, 1997).

LSA is able to relate coherent contexts to specific content words, and it is good at predicting the occurrence of a content word in the presence of other thematically related terms. However, since it does not take word order into account (“bag-of-words” model) it is very poor at predicting their actual position within the sentence, and it is completely useless for the prediction of function words. Therefore, some attempts have been made to integrate the information coming from an LSA-based model with standard language models of the n-gram type.

In the LSA model (Deerwester et al, 1990) a word w_i is represented as a high-dimensional vector, derived by *Singular Value Decomposition* (SVD) from a term × document (or a term × term) co-occurrence matrix of a training corpus. In this framework, a context or history h (= $w_1, \dots , w_m$) can be represented by the sum of the (already normalized) vectors corresponding to the words it contains (Landauer et al. 1997):

$$\vec{h} = \sum_{i=1}^{m} \vec{w}_i \qquad (2)$$

This vector reflects the meaning of the preceding (already typed) section, and it has the same dimensionality as the term vectors. It can thus be compared to the term vectors by well-known similarity measures (scalar product, cosine).

2.3 Transforming LSA similarities into probabilities

We make the assumption that an utterance or a text to be entered is usually semantically cohesive. We then expect all word vectors to be close to the current context vector, whose corresponding words belong to the semantic field of the context. This forms the basis for a simple probabilistic model of LSA: After calculating the cosine similarity for each word vector $\vec{w}_i$ with the vector $\vec{h}$ of the current context, we could use the normalized similarities as probability values. This probability distribution however is usually rather flat (i.e. the dynamic range is low). For this reason a contrasting (or temperature) factor γ is normally applied (cf. Coccaro and Jurafsky, 1998), which raises the cosine to some power (γ is normally between 3 and 8). After normalization we obtain a probability distribution which can be used for prediction purposes. It is calculated as follows:

$$P_{LSA}(w_i|h) = \frac{\left(\cos(\vec{w}_i,\vec{h}) - \cos_{\min}(\vec{h})\right)^{\gamma}}{\sum_k \left(\cos(\vec{w}_k,\vec{h}) - \cos_{\min}(\vec{h})\right)^{\gamma}} \qquad (3)$$

w_i is a word in the vocabulary, h is the current context (history) $\vec{w}_i$ and $\vec{h}$ are their corresponding vectors in the LSA space; cos_{min}($\vec{h}$) returns the lowest cosine value measured for $\vec{h}$). The denominator then normalizes each similarity value to ensure that $\sum_k^n P_{LSA}(w_k,h) = 1$.

Let us illustrate the capacities of this model by giving a short example from the French version of our own LSA predictor:

Context: *"Mon père était professeur en mathématiques et je pense que "*
("My dad has been a professor in mathematics and I think that ")

Rank	Word	P
1.	*professeur* ('professor')	0.0117
2.	*mathématiques* ("mathematics")	0.0109
3.	*enseigné* (participle of 'taught')	0.0083
4.	*enseignait* ('taught')	0.0053
5.	*mathematicien* ('mathematician')	0.0049
6.	*père* ('father')	0.0046
7.	*mathématique* ('mathematics')	0.0045
8.	*grand-père* ('grand-father')	0.0043
9.	*sciences* ('sciences')	0.0036
10.	*enseignant* ('teacher')	0.0032

Example 1: Most probable words returned by the LSA model for the given context.

As can be seen in example 1, all ten predicted words are semantically related to the context, they should therefore be given a high probability of occurrence. However, this example also shows the drawbacks of the LSA model: it totally neglects the presence of function words as well as the syntactic structure of the current phrase. We therefore need to find an appropriate way to integrate the information coming from a standard n-gram model and the LSA approach.

2.4 Density as a confidence measure

Measuring relation quality in an LSA space, Wandmacher (2005) pointed out that the reliability of LSA relations varies strongly between terms. He also showed that the entropy of a term does not correlate with relation quality (i.e. number of semantically related terms in an LSA-generated term cluster), but he found a medium correlation (*Pearson* coeff. = 0.56) between the number of semantically related terms and the average cosine similarity of the *m* nearest neighbors (density). The closer the nearest neighbors of a term vector are, the more probable it is to find semantically related terms for the given word. In turn, terms having a high density are more likely to be semantically related to a given context (i.e. their specificity is higher).

We define the density of a term w_i as follows:

$$D_m(w_i) = \frac{1}{m} \cdot \sum_{j=1}^{m} \cos(\vec{w}_i, NN_j(\vec{w}_i)) \qquad (4)$$

In the following we will use this measure (with *m*=100) as a confidence metric to estimate the reliability of a word being predicted by the LSA component, since it showed to give slightly better results in our experiments than the entropy measure.

3 Integrating semantic information

In the following we present several different methods to integrate semantic information as it is provided by an LSA model into a standard LM.

3.1 Semantic cache model

Cache (or recency promotion) models have shown to bring slight but constant gains in language modeling (Kuhn and De Mori, 1990). The underlying idea is that words that have already occurred in a text are more likely to occur another time. Therefore their probability is raised by a constant or exponentially decaying factor, depending on the position of the element in the cache. The idea of a decaying cache function is that the probability of reoccurrence depends on the cosine similarity of the word in the cache and the word to be predicted. The highest probability of reoccurrence is usually after 15 to 20 words.

Similar to Clarkson and Robinson (1997), we implemented an exponentially decaying cache of length *l* (usually between 100 and 1000), using the following decay function for a word w_i and its position p in the cache.

$$f_d(w_i, p) = e^{\left(\frac{-0,5(p-\mu)}{\sigma}\right)^2} \qquad (5)$$

$\sigma = \mu/3$ if $p < \mu$ and $\sigma = l/3$ if $p \geq \mu$. The function returns 0 if w_i is not in the cache, and it is 1 if $p = \mu$. A typical graph for (5) can be seen in figure (2).

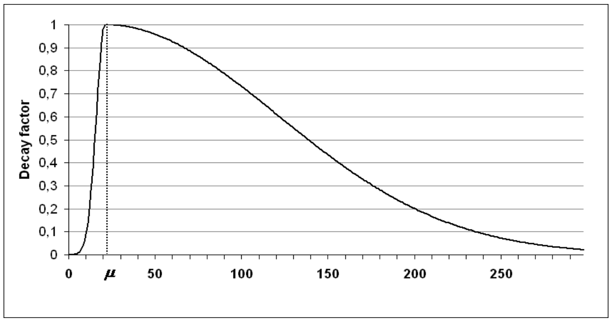


Figure 2: Decay function with μ=20 and l=300.

We extend this model by calculating for each element having occurred in the context its m nearest LSA neighbors ($NN_m(\vec{w}_{occ}, \theta)$, using cosine similarity), if their cosine lies above a threshold θ, and add them to the cache as well, right after the word that has occurred in the text ("*Bring your friends*"-strategy). The size of the cache is adapted accordingly (for μ, σ and l), depending on the number of neighbors added. This results in the following cache function:

$$P_{cache}(w_i) = \sum\nolimits_1^l \beta \cdot f_{\cos}(w^i_{occ}, w_i) \cdot f_d(w_i, p) \qquad (6)$$

with l = size of the cache. β is a constant controlling the influence of the component (usually $\beta \approx 0.1/l$); w^i_{occ} is a word that has already recently occurred in the context and is therefore added as a standard cache element, whereas w_i is a nearest neighbor to w^i_{occ}. $f_{cos}(w^i_{occ}, w_i)$ returns the cosine similarity between $\vec{w}^i_{occ}$ and $\vec{w}_i$, with $cos(\vec{w}^i_{occ}, \vec{w}_i) > \theta$ (Rem: w_i with $cos(\vec{w}^i_{occ}, \vec{w}_i) \leq \theta$ have not been added to the cache). Since $cos(\vec{w}_i, \vec{w}_i)$=1, terms having actually occurred before will be given full weight, whereas all w_i being only nearest LSA neighbors to w^i_{occ} will receive a weight according to their cosine similarity with w^i_{occ}, which is less than 1 (but larger than θ).

$f_d(w_i,p)$ is the decay factor for the current position p of w_i in the cache, calculated as shown in equation (5).

3.2 Partial reranking

The underlying idea of partial reranking is to regard only the best n candidates from the basic language model for the semantic model in order to prevent the LSA model from making totally implausible (i.e. improbable) predictions. Words being improbable for a given context will be disregarded as well as words that do not occur in the semantic model (e.g. function words), because LSA is not able to give correct estimates for this group of words (here the base probability remains unchanged).

For the best n candidates their semantic probability is calculated and each of these words is assigned an additional value, after a fraction of its base probability has been subtracted (*jackpot* strategy).

For a given context h we calculate the ordered set

BEST$_n(h)$ = $<w_1, \ldots, w_n>$, so that $P(w_1|h) \geq P(w_2|h) \geq \ldots \geq P(w_n|h)$

For each w_i in BEST$_n(h)$ we then calculate its reranking probability as follows:

$$P_{RR}(w_i) = \beta \cdot \cos(\vec{w}_i, \vec{h}) \cdot D(w_i) \cdot I(Best_n(h), w_i) \quad (7)$$

β is a weighting constant controlling the overall influence of the reranking process, $cos(\vec{w}_i, \vec{w}_i)$ returns the cosine of the word's vector and the current context vector, $D(w_i)$ gives the confidence measure of w_i and I is an indicator function being 1, iff $w_i \in$ BEST(h), and 0 otherwise.

3.3 Standard interpolation

Interpolation is the standard way to integrate information from heterogeneous resources. While for a linear combination we simply add the weighted probabilities of two (or more) models, geometric interpolation multiplies the probabilities, which are weighted by an exponential coefficient ($0 \leq \lambda_1 \leq 1$):

Linear Interpolation (LI):

$$P'(w_i) = \lambda_1 \cdot P_b(w_i) + (1 - \lambda_1) \cdot P_s(w_i) \quad (8)$$

Geometric Interpolation (GI):

$$P'(w_i) = \frac{P_b(w_i)^{\lambda_1} \cdot P_s(w_i)^{(1-\lambda_1)}}{\sum_{j=1}^{n} P_b(w_j)^{\lambda_1} \cdot P_s(w_j)^{(1-\lambda_1)}} \quad (9)$$

The main difference between the two methods is that the latter takes the agreement of two models into account. Only if each of the single models assigns a high probability to a given event will the combined probability be assigned a high value. If one of the models assigns a high probability and the other does not the resulting probability will be lower.

3.4 Confidence-weighted interpolation

Whereas in standard settings the coefficients are stable for all probabilities, some approaches use confidence-weighted coefficients that are adapted for each probability. In order to integrate n-gram and LSA probabilities, Coccaro and Jurafsky (1998) proposed an entropy-related confidence measure for the LSA component, based on the observation that words that occur in many different contexts (i.e. have a high entropy), cannot well be predicted by LSA. We use here a density-based measure (cf. section 2.2), because we found it more reliable than entropy in preliminary tests. For interpolation purposes we calculate the coefficient of the LSA component as follows:

$$\lambda_i = \beta \cdot D(w_i)\text{, iff } D(w_i) > 0\text{; 0 otherwise} \quad (10)$$

with β being a weighting constant to control the influence of the LSA predictor. For all experiments, we set β to 0.4 (i.e. $0 \le \lambda_i \le 0.4$), which proved to be optimal in pre-tests.

4 Results

We calculated our baseline n-gram model on a 44 million word corpus from the French daily *Le Monde* (1998-1999). Using the *SRI* toolkit (Stolcke, 2002)[1] we computed a 4-gram LM over a controlled 141,000 word vocabulary, using *modified Kneser-Ney* discounting (Goodman, 2001), and we applied *Stolcke* pruning (Stolcke, 1998) to reduce the model to a manageable size ($\theta = 10^{-7}$).

The LSA space was calculated on a 100 million word corpus from *Le Monde* (1996 – 2002). Using the *Infomap* toolkit[2], we generated a term × term co-occurrence matrix for an 80,000 word vocabulary (matrix size = 80,000 × 3,000), stopwords were excluded. After several pre-tests, we set the size of the co-occurrence window to ±100. The matrix was then reduced by singular value decomposition to 150 columns, so that each word in the vocabulary was represented by a vector of 150 dimensions, which was normalized to speed up similarity calculations (the scalar product of two normalized vectors equals the cosine of their angle).

Our test corpus consisted of 8 sections from the French newspaper *Humanité*, (January 1999, from 5,378 to 8,750 words each), summing up to 58,457 words. We then calculated for each test set the keystroke saving rate based on a 5-word list (ksr_5) and perplexity for the following settings[3]:

1. 4-gram LM only (baseline)
2. 4-gram + decaying cache (l = 400)
3. 4-gram + LSA using linear interpolation with λ_{LSA} = 0.11 (LI).
4. 4-gram + LSA using geometric interpolation, with λ_{LSA} = 0.07 (GI).
5. 4-gram + LSA using linear interpolation and (density-based) confidence weighting (CWLI).
6. 4-gram + LSA using geometric interpolation and (density-based) confidence weighting (CWGI).
7. 4-gram + partial reranking (n = 1000, β = 0.001)
8. 4-gram + decaying semantic cache (l = 4000; m = 10; θ = 0.4, β = 0.0001)

Figures 3 and 4 display the overall results in terms of *ksr* and perplexity.

[1] SRI Toolkit: www.speech.sri.com.

[2] Infomap Project: http://infomap-nlp.sourceforge.net/

[3] All parameter settings presented here are based on results of extended empirical pre-tests. We used held-out development data sets that have randomly been chosen from the *Humanité* corpus.(8k to 10k words each). The parameters being presented here were optimal for our test sets. For reasons of simplicity we did not use automatic optimization techniques such as the EM algorithm (cf. Jelinek, 1990).

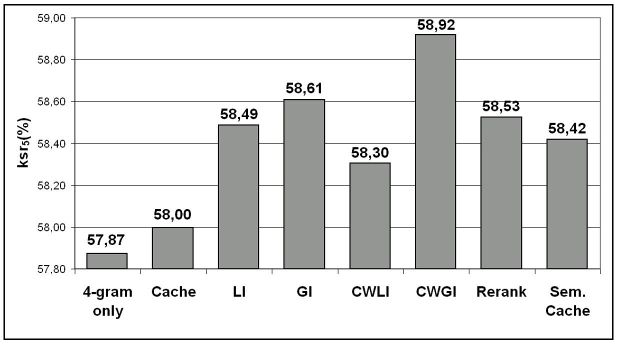


Figure 3: Results (*ksr*$_5$) for all methods tested.

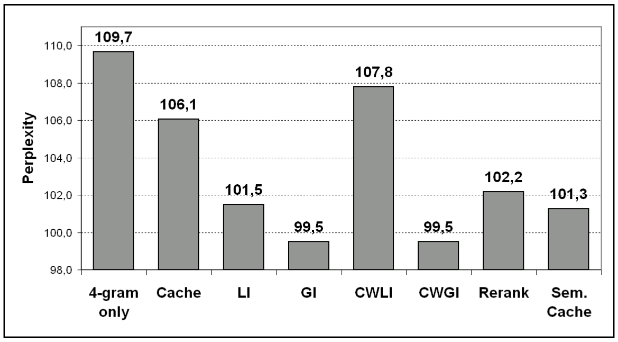


Figure 4: Results (perplexity) for all methods tested.

Using the results of our 8 samples, we performed paired *t* tests for every method with the baseline as well as with the cache model. All gains for *ksr* turned out to be highly significant (sig. level < 0.001), and apart from the results for CWLI, all perplexity reductions were significant as well (sig. level < 0.007), with respect to the cache results. We can therefore conclude that, with exception of CWLI, all methods tested have a beneficial effect, even when compared to a simple cache model. The highest gain in *ksr* (with respect to the baseline) was obtained for the confidence-weighted geometric interpolation method (CWGI; +1.05%), the highest perplexity reduction was measured for GI as well as for CWGI (-9.3% for both). All other methods (apart from IWLI) gave rather similar results (+0.6 to +0.8% in *ksr*, and -6.8% to -7.7% in perplexity).

We also calculated for all samples the correlation between *ksr* and perplexity. We measured a *Pearson* coefficient of -0.683 (Sig. level < 0.0001).

At first glance, these results may not seem overwhelming, but we have to take into account that our *ksr* baseline of 57.9% is already rather high, and at such a level, additional gains become hard to achieve (cf. Lesher et al, 2002).

The fact that CWLI performed worse than even simple LI was not expected, but it can be explained by an inherent property of linear interpolation: If one of the models to be interpolated overestimates the probability for a word, the other cannot compensate for it (even if it gives correct estimates), and the resulting probability will be too high. In our case, this happens when a word receives a high confidence value; its probability will then be overestimated by the LSA component.

5 Conclusion and further work

Adapting a statistical language model with semantic information, stemming from a distributional analysis like LSA, has shown to be a non-trivial problem. Considering the task of word prediction in an AAC system, we tested different methods to integrate an n-gram LM with LSA: A semantic cache model, a partial reranking approach, and some variants of interpolation.

We evaluated the methods using two different measures, the keystroke saving rate (*ksr*) and perplexity, and we found significant gains for all methods incorporating LSA information, compared to the baseline. In terms of *ksr* the most successful method was confidence-weighted geometric interpolation (CWGI; +1.05% in *ksr*); for perplexity, the greatest reduction was obtained for standard as well as for confidence-weighted geometric interpolation (-9.3% for both). Partial reranking and the semantic cache gave very similar results, despite their rather different underlying approach.

We could not provide here a comparison with other models that make use of distributional information, like the trigger approach by Rosenfeld (1996), Matiasek and Baroni (2003) or the model presented by Li and Hirst (2005), based on *Pointwise Mutual Information* (PMI). A comparison of these similarities with LSA remains to be done.

Finally, an AAC system has not only the function of simple text entering but also of providing cognitive support to its user, whose communicative abilities might be totally depending on it. Therefore, she or he might feel a strong improvement of the system, if it can provide semantically plausible predictions, even though the actual gain in *ksr* might be modest or even slightly decreasing. For this reason we will perform an extended qualitative analysis of the presented methods with persons who use our AAC system *SIBYLLE*. This is one of the main aims of the recently started *ESAC_IMC* project. It is conducted at the *Functional Reeducation and Rehabilitation Centre* of Kerpape, Brittany, where *SIBYLLE* is already used by 20 children suffering from traumatisms of the motor cortex. They appreciate the system not only for communication but also for language learning purposes.

Moreover, we intend to make the word predictor of *SIBYLLE* publicly available (*AFM Voltaire project*) in the not-too-distant future.

Acknowledgements

This research is partially founded by the UFA (Université Franco-Allemande) and the French foundations APRETREIMC (*ESAC_IMC* project) and AFM (VOLTAIRE project). We also want to thank the developers of the *SRI* and the *Infomap* toolkits for making their programs available.

References

Bellegarda, J. (1997): “A Latent Semantic Analysis Framework for Large-Span Language Modeling”, *Proceedings of the Eurospeech 97,* Rhodes, Greece.

Boissière Ph. and Dours D. (1996). “VITIPI : Versatile interpretation of text input by persons with impairments”. *Proceedings ICCHP'1996.* Linz, Austria.

Church, K. and Hanks, P. (1989). “Word association norms, mutual information and lexicography”. *Proceedings of ACL*, pp. 76-83.

Clarkson, P. R. and Robinson, A.J. (1997). “Language Model Adaptation using Mixtures and an Exponentially Decaying Cache”, in *Proc. of the IEEE ICASSP-97*, Munich.

Coccaro, N. and Jurafsky, D. (1998). “Towards better integration of semantic predictors in statistical language modeling”, *Proc. of the ICSLP-98*, Sydney.

Deerwester, S. C., Dumais, S., Landauer, T., Furnas, G. and Harshman, R. (1990). “Indexing by Latent Semantic Analysis”, *JASIS* 41(6), pp. 391-407.

Fazly, A. and Hirst, G. (2003). “Testing the efficacy of part-of-speech information in word completion”, *Proceedings of the Workshop on Language Modeling for Text Entry Methods on EACL*, Budapest.

Goodman, J. (2001): “A Bit of Progress in Language Modeling”, Extended Version Microsoft Research Technical Report MSR-TR-2001-72.

Jelinek, F. (1990): “Self-organized Language Models for Speech Recognition”, In: A. Waibel and K.-F. Lee (eds.), *Readings in Speech Recognition*, Morgan Kaufman Publishers, pp. 450-506.

Kuhn, R. and De Mori, R. (1990). “A Cache-Based Natural Language Model for Speech Reproduction”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (6), pp. 570-583.

Landauer, T. K., Laham, D., Rehder, B. and Schreiner, M. E. (1997). “How well can passage meaning be derived without using word order? A comparison of LSA and humans”, *Proceedings of the 19th annual meeting of the Cognitive Science Society*, pp. 412-417, Erlbaum Mawhwah, NJ.

Lesher, G. W., Moulton, B. J, Higginbotham, D.J. and Alsofrom, B. (2002). “Limits of human word prediction performance”, *Proceedings of the CSUN 2002.*

Li, J., Hirst, G. (2005). “Semantic knowledge in a word completion task”, *Proc. of the 7th Int. ACM Conference on Computers and Accessibility*, Baltimore.

Matiasek, H. and Baroni, M. (2003). “Exploiting long distance collocational relations in predictive typing”, *Proceedings of the EACL-03 Workshop on Language Modeling for Text Entry Methods*, Budapest.

Rosenfeld, R. (1996). “A maximum entropy approach to adaptive statistical language modelling”, *Computer Speech and Language*, 10 (1), pp. 187-228.

Schadle I., Antoine J.-Y., Le Pévédic B., Poirier F. (2004). “Sibyl - AAC system using NLP techniques”. Proc. *ICCHP’2004*, Paris, France. *LNCS 3118*, Springer Verlag.

Stolcke, A. (1998): “Entropy-based pruning of backoff language models”. *Proc.s of the DARPA Broadcast News Transcription and Understanding Workshop.*

Stolcke, A. (2002): “SRILM - An Extensible Language Modeling Toolkit”, in *Proc. of the Intl. Conference on Spoken Language Processing*, Denver, Colorado.

Trnka, K., Yarrington, D., McCoy, K. F. and Pennington, C. (2006): “Topic Modeling in Fringe Word Prediction for AAC”, In *Proceedings of the 2006 International Conference on Intelligent User Interfaces*, pp. 276 – 278, Sydney, Australia.

Trost, H., Matiasek, J. and Baroni, M. (2005): “The Language Component of the FASTY Text Prediction System”, *Applied Artificial Intelligence*, 19 (8), pp. 743-781.

Wandmacher, T. (2005): “How semantic is Latent Semantic Analysis?”, in *Proceedings of TALN/RECITAL 2005*, Dourdan, France, 6-10 june.

Bilingual Cluster Based Models for Statistical Machine Translation

Hirofumi Yamamoto
National Institute of Information
and Communications Technology
/ 2-2-2 Hikaridai Seika-cho
Soraku-gun Kyoto Japan
and ATR Spoken Language
Communication Research Labs.
hirofumi.yamamoto@nict.go.jp

Eiichiro Sumita
National Institute of Information
and Communications Technology
/ 2-2-2 Hikaridai Seika-cho
Soraku-gun Kyoto Japan
and ATR Spoken Language
Communication Research Labs.
eiichiro.sumita@nict.go.jp

Abstract

We propose a domain specific model for statistical machine translation. It is well-known that domain specific language models perform well in automatic speech recognition. We show that domain specific language and translation models also benefit statistical machine translation. However, there are two problems with using domain specific models. The first is the data sparseness problem. We employ an adaptation technique to overcome this problem. The second issue is domain prediction. In order to perform adaptation, the domain must be provided, however in many cases, the domain is not known or changes dynamically. For these cases, not only the translation target sentence but also the domain must be predicted. This paper focuses on the domain prediction problem for statistical machine translation. In the proposed method, a bilingual training corpus, is automatically clustered into sub-corpora. Each sub-corpus is deemed to be a domain. The domain of a source sentence is predicted by using its similarity to the sub-corpora. The predicted domain (sub-corpus) specific language and translation models are then used for the translation decoding. This approach gave an improvement of 2.7 in BLEU (Papineni et al., 2002) score on the IWSLT05 Japanese to English evaluation corpus (improving the score from 52.4 to 55.1). This is a substantial gain and indicates the validity of the proposed bilingual cluster based models.

1 Introduction

Statistical models, such as n-gram models, are widely used in natural language processing, for example in speech recognition and statistical machine translation (SMT). The performance of a statistical model has been shown to improve when domain specific models are used, since similarity of statistical characteristics between model and target is higher. For utilize of domain specific models, a training data sparseness and target domain estimation problems must be resolved. In this paper, we try to estimate target domain sentence by sentence, considering cases where the domain changes dynamically. After sentence by sentence domain estimation, domain specific models are used for translation using the adaptation technique(Seymore et al., 1997).

In order to train a classifier to predict the domain, we used an unsupervised clustering technique on an unlabelled bilingual training corpus. We regarded each cluster (sub-corpus) as a domain. Prior to translation, the domain of the source sentence is first predicted and this prediction is then used for model selection. The most similar sub-corpus to the translation source sentence is used to represent its domain. After the prediction is made, domain specific language and translation models are used for the translation.

In Section 2 we present the formal basis for our domain specific translation method. In Section 3 we provide a general overview of the two sub-tasks of domain specific translation: domain prediction, and domain specific decoding. Section 4 presents the domain prediction task in depth. Section 5 offers a more detailed description of the details of domain specific decoding. Section 6 gives details of the experiments and presents the results. Finally, Section

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 514–523, Prague, June 2007. ©2007 Association for Computational Linguistics

7 offers a summary and some concluding remarks.

2 Domain Specific Models in SMT

The purpose of statistical machine translation is to find the most probable translation in the target language e of a given source language sentence f. This search process can be expressed formally by:

$$\underset{e}{argmax}\, P(e|f) \tag{1}$$

In this formula, the target word sequence (sentence) e is determined only by the source language word sequence f. However, e is heavily dependent on not only on f but also on the domain D. When the domain D is given, formula (1) can be rewritten as the following formula with the introduction of a new probabilistic variable D.

$$\underset{e}{argmax}\, P(e|f, D) \tag{2}$$

This formula can be re-expressed using Bayes' Law.

$$\underset{e}{argmax}\, P(e|D)P(f|e, D) \tag{3}$$

Here, $P(f|e, D)$ represents the domain D specific translation model and $P(e|D)$ represents the domain D specific language model.

When the domain D is known, domain specific models can be created and used in the translation decoding process. However, in many cases, domain D is unknown or changes dynamically. In these cases, both the translation target language sentence e and the domain D must be dynamically predicted at the same time. The following equation represents the process of domain specific translation when the domain D is being dynamically predicted.

$$\begin{aligned} & \underset{e,D}{argmax}\, P(e, D|f) \\ = & \underset{e,D}{argmax}\, P(D|f)P(e|f, D) \end{aligned} \tag{4}$$

The major difference between this equation and formula (3) is that the probabilistic variable D is the prediction target in equation (4). In this equation, $P(D|f)$ represents the domain prediction and $P(e|f, D)$ represents the domain specific translation.

3 Outline of the Proposed Method

Our method can be analysed into two processes: an off-line process and an on-line process. The processes are depicted in figure 1. In the off-line process, bilingual sub-corpora are created by clustering and these clusters represent domains. Domain specific models are then created from the data contained in the sub-corpora in a batch process. In the on-line process, the domain of the source sentence is first predicted and following this the sentence is translated using models built on data from the appropriate domain.

3.1 Off-line process

In this process, the training corpus is clustered to sub-corpora, which are regarded as domains. In SMT, a bilingual corpus is used to create the translation model, and typically, bilingual data together with additional monolingual corpora are used to create the language model. In our method, both the bilingual and monolingual corpora are clustered. After clustering, cluster dependent (domain specific) language and translation models are created from the data in the clusters.

1. A bilingual corpus which is comprised of the training data for the translation model, or equivalently the bilingual part of the training data for the language model is clustered (see Section 4.2).

2. Each sentence of the additional monolingual corpora (if any) is assigned to a bilingual cluster (see Section 4.3).

3. For each cluster, the domain specific (cluster dependent) language models are created.

4. The domain specific translation model is created using only the clusters formed from clustering bilingual data.

3.2 On-line process

This process is comprised of domain prediction and the domain specific translation components. The following steps are taken for each source sentence.

1. Select the cluster to which the source sentence belongs.

2. Translate the source sentence using the appropriate domain specific language and translation models.

4 Domain Prediction

This section details the domain prediction process. To satisfy equation (4), both the domain D and the translation target word sequence e, which maximizes both $P(D|f)$ and $P(e|f, D)$ must be calculated at the same time. However, it is difficult to make the calculations without an approximation. Therefore, in the first step, we find the best candidates for D given the input sentence f. In the next step, $P(e|f, D)$ is maximized over the candidates for D using the following formula.

$$\underset{e}{argmax}\ P(e|f, \underset{D}{argmax}\ P(D|f)) \tag{5}$$

Equation (5) is approximation of following equation in that can D is regarded as a hidden variable.

$$\underset{e}{argmax} \sum_{D} P(D|f)P(e|D)P(f|e, D)) \tag{6}$$

When the following assumptions are introduced to equation (6), equation (5) is obtained as an approximation. For only one domain D_i, $P(D_i|f)$ is nearly equal to one. For other domains, $P(D|f)$ are almost zero. $P(D|f)$ can be re-written as following equation.

$$\begin{aligned} & P(D|f) \\ = \ & P(D, f)/P(f) \\ = \ & P(f, D)/P(D) \times P(D)/P(f) \\ = \ & P(f|D)P(D)/P(f) \end{aligned} \tag{7}$$

Therefore, we can confirm reasonability of this assumption by calculating $P(f|D)P(D)$ all domains ($P(f)$ is constant).

4.1 Domain Definition

When the domain is known in advance, it is usually expressible, for example it could be a topic that matches a human-defined category like “sport”. On the other hand, when the domain is delimited in an unsupervised manner, it is used only as a probabilistic variable and does not need to be expressed. Equation (4) illustrates that a good model will provide high probabilities to $P(D|f)P(e|f, D)$ for bilingual sentence pairs (f, e). For the same reason, a good domain definition will lead to a higher probability for the term: $P(D|f)P(e|f, D)$. Therefore, we define the domain D as that which maximizes $P(D|f)P(e|D)$ (an approximation of $P(D|f)P(e|f, D)$). This approximation ensures that the domain definition is optimal for only the language model rather than both the language and translation models. $P(D|f)P(e|D)$ can be re-written as the following equation using Bayes’ Law.

$$\begin{aligned} & P(D|f)P(e|D) \\ = \ & P(e|D)P(f|D)P(D)/P(f) \end{aligned} \tag{8}$$

Here, $P(f)$ is independent of domain D. Furthermore, we assume $P(D)$ to be constant. The following formula embodies the search for the optimal domain.

$$\underset{D}{argmax}\ P(e|D)P(f|D) \tag{9}$$

This formula ensures that the search for the domain maximizes the domain specific probabilities of both e and f simultaneously.

4.2 Clustering of the bilingual corpus

As mentioned above, we maximize the domain specific probabilities of e and f to ascertain the domain. We define our domains as sub-corpora of the bilingual corpus, and these sub-corpora are formed by clustering bilingually by entropy reduction. For this clustering, the following extension of monolingual corpus clustering is employed (Carter 1994).

1. The total number of clusters (domains) is given by the user.

2. Each bilingual sentence pair is randomly assigned to a cluster.

3. For each cluster, language models for e and f are created using the bilingual sentence pairs that belong to the cluster.

4. For each cluster, the entropy for e and f is calculated by applying the language models from the previous step to the sentences in the cluster. The total entropy is defined as the total sum of entropy (for both source and target) for each cluster.

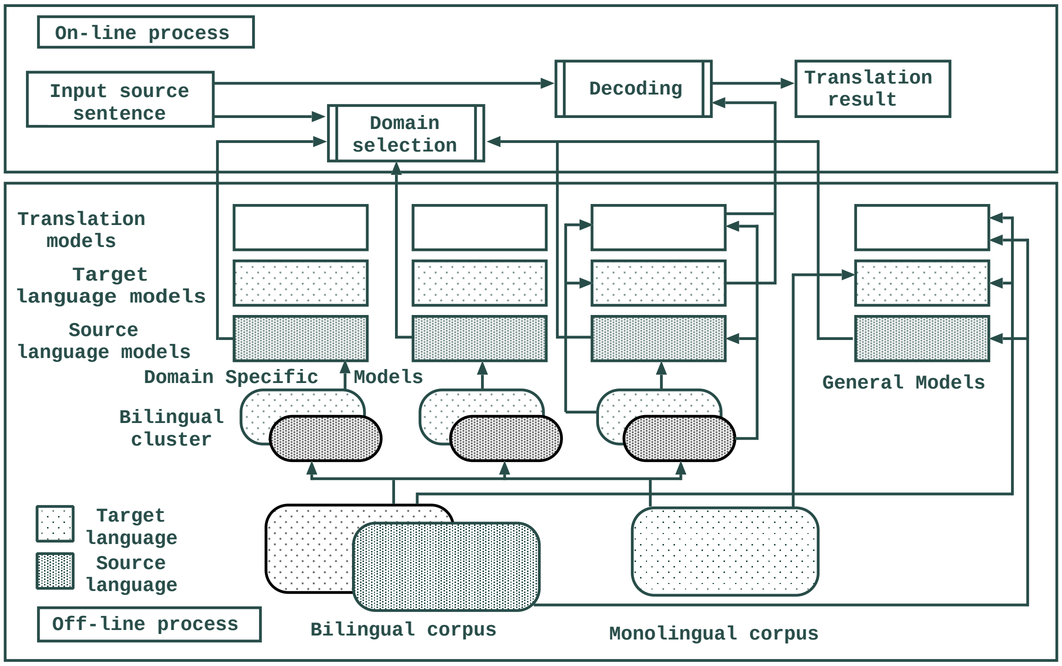


Figure 1: Outline of the Proposed Method

5. Each bilingual sentence pair is re-assigned to a cluster such that the assignment minimizes the total entropy.

6. The process is repeated from step (3) until the entropy reduction is smaller than a given threshold.

4.3 Clustering the monolingual corpus

Any additional monolingual corpora used to train the language model are also clustered. For this clustering, the following process is used.

1. First, bilingual clusters are created using the above process.

2. For each monolingual sentence its entropy is calculated using all the bilingual cluster dependent language models and also the general language model (see Figure 1 for a description of the general language model).

3. If the entropy of the general language model is the lowest, this sentence is not used in the cluster dependent language models.

4. Otherwise, the monolingual sentence is added to the bilingual cluster that results in the lowest entropy.

4.4 Domain prediction

In the process described in the previous section we describe how clusters are created, and we define our domains in terms of these clusters. In this step, domain D is predicted using the given source sentence f. This prediction is equivalent to finding the D that maximizes $P(D|f)$. $P(D|f)$ can be re-written as $P(f|D)P(D)/P(f)$ using Bayes' law. Here, $P(f)$ is a constant, and if $P(D)$ is assumed to be constant (this approximation is also used in the clustering of the bilingual corpus), maximizing the target is reduced to the maximization of $P(f|D)$. To maximize $P(f|D)$ we simply select the cluster D, that gives the highest likelihood of a given source sentence f.

5 Domain specific decoding

After domain prediction, domain specific decoding to maximize $P(e|f,D)$, is conducted. $P(e|f,D)$ can be re-written as the following equation using Bayes' law.

$$\begin{aligned} & P(e|f,D) \\ = \ & P(f|e,D)P(e,D)/P(f,D) \end{aligned}$$

$$= \quad P(f|e,D)P(e|D)P(D)/P(f,D) \quad (10)$$

Here, f is a given constant and D has already been selected by the domain prediction process. Therefore, maximizing $P(f|e,D)P(e|D)$ is equivalent to maximizing the above equation. In $P(f|e,D)P(e|D)$, $P(f|e,D)$ is the domain specific translation model and $P(e|D)$ is the domain specific language model. Equation (10) represents the whole process of translation of f into e using domain D specific models $P(f|e,D)$ and $P(e|D)$.

5.1 Differences from previous methods

5.1.1 Cluster language model

Hasan et al. (2005) proposed a cluster language model for finding the domain D. This method has three steps. In the first step, the translation target language corpus is clustered using human-defined regular expressions. In the second step, a regular expression is created from the source sentence f. In the last step, the cluster that corresponds to the extracted regular expression is selected, and the cluster specific language model built from the data in this cluster is used for the translation. The points of difference are:

- In the cluster language model, clusters are defined by human-defined regular expressions. On the other hand, with the proposed method, clusters are automatically (without human knowledge) defined and created by the entropy reduction based method.
- In the cluster language model, only the translation target language corpus is clustered. In the proposed method, both the translation source and target language corpora are clustered (bilingual clusters).
- In the cluster language model, only a domain (cluster) specific language model is used. In the proposed method, both a domain specific language model and a domain specific translation model are used.

5.1.2 Sentence mixture language model

In equation (6), D is regarded as a hidden variable. Furthermore, when $P(D|f)$ is approximated as $P(D) = D_\lambda$, and the general translation model $P(f|e)$ is used instead of the domain specific translation model $P(f|e,D)$, this equation represents the process of translation using sentence mixture language models (Iyer et al., 1993) as follows:

$$\underset{e}{argmax} \sum_D D_\lambda P(e|D)P(f|e) \qquad (11)$$

The points that differ from the proposed method are as follows:

- In the sentence mixture model, the mixture weight parameters D_λ are constant. On the other hand, in the proposed method, weight parameters $P(D|f)$ are estimated separately for each sentence.
- In the sentence mixture model, the probabilities of all cluster dependent language models are summed. In the proposed model, only the cluster that gives the highest probability is considered as approximation.
- In the proposed method, a domain specific translation model is also used.

6 Experiments

6.1 Japanese to English translation

6.1.1 Experimental corpus

To evaluate the proposed model, we conducted experiments based on a travel conversation task corpus. The experimental corpus was the travel arrangements task of the BTEC corpus (Takezawa et al., 2002),(Kikui et al., 2003) and the language pair was Japanese and English. The training, development, and evaluation corpora are shown in Table 1. The development and evaluation corpora each had sixteen reference translations for each sentence. This training corpus was also used for the IWSLT06 Evaluation Campaign on Spoken Language Translation (Paul 2006) J-E open track, and the evaluation corpus was used as the IWSLT05 evaluation set.

6.1.2 Experimental conditions

For bilingual corpus clustering, the sentence entropy must be calculated. Unigram language models were used for this calculation. The translation models were pharse-based (Zen et al., 2002) created using the GIZA++ toolkit (Och et al., 2003). The language models for the domain prediction and translation decoding were word trigram with Good-Turing

Table 1: Japanese to English experimental corpus

	# of sentence	Total words	# of word entry
Japanese Training	40K	355K	12.5K
English Training	40K	315K	9.2K
Japanese Development	510	3,525	918
English Development	510×16	57,388	2,118
Japanese Evaluation	506	3,647	951

backoff (Katz 1987). Ten cluster specific source language models and a general language model were used for the domain prediction. If the general language model provided the lowest perplexity for an input sentence, the domain specific models were not used for this sentence. The SRI language modeling toolkit (Stolcke) was used for the creation of all language models. The PHARAOH phrase-based decoder (Koehn 2004) was used for the translation decoding.

For tuning of the decoder's parameters, including the language model weight, minimum error training (Och 2003) with respect to the BLEU score using was conducted using the development corpus. These parameters were used for the baseline conditions. During translation decoding, the domain specific language model was used as an additional feature in the log-linear combination according to the PHARAOH decoder's option. That is, the general and domain specific language models are combined by log-linear rather than linear interpolation. The weight parameters for the general and domain specific language models were manually tuned using the development corpus. The sum of these language model weights was equal to the language model weight in the baseline. For the translation model, the general translation model (phrase table) and domain specific translation model were linearly combined. The interpolation parameter was again manually tuned using the development corpus.

6.1.3 Experimental results

In our bilingual clustering, the number of clusters must be fixed in advance. Based on the results of preliminary experiments to estimate model order, ten clusters were used. If less than ten clusters were used, domain specific characteristics cannot be represented. If more than ten clusters were used, data sparseness problems are severe, especially in translation models. The amount of sentences in each cluster is not so different, therefore the approximation that $P(D)$ is reasonable. Two samples of bilingual clusters are recorded in the appendix "Sample of Cluster". The cluster A.1 includes many interrogative sentences. The reason is that special words "ですか (desu ka)" or "ますか (masu ka)" are used at the end of Japanese sentence with no corresponding word used in English. The cluster A.2 includes numeric expressions in both English and Japanese.

Next, we confirm the reasonability of the assumption used in equation(5). For this confirmation, we calculate $P(D|f)$ for all D for each f (P(D) is approximated as constant). For almost f, only one domain D_i has a vary large value compared with other domains. Therefore, this approximation is confirmed to be reasonable.

In this experiments, we compare three ways of deploying our domain specific models to a baseline. In the first method, only the domain specific language model is used. The ratio of the weight parameter for the general model to the domain specific model was 6:4 for all the domain specific language models. In the second method, only the domain specific translation model was used. The ratio of the interpolation parameter of the general model to the domain specific model was 3:7 for all the domain specific models. In the last method, both the domain specific language and translation models (LM+TM) were used. The weights and interpolation parameters were the same as in the first and second methods. The experimental results are shown in Table 2. Under all of the conditions and for all of the evaluation measures, the proposed domain specific models gave better performance than the baseline. The highest performance came from the system that used both the domain specific language and translation models, resulting in a

2.7 point BLEU score gain over the baseline. It is a very respectable improvement. Appendix ”Sample of Different Translation Results” recodes samples of different translation results with and without the domain specific language and translation models. In many cases, better word order is obtained in with the domain specific models.

6.2 Translation of ASR output

In this experiment, the source sentence used as input to the machine translation system was the direct textual output from an automatic speech recognition (ASR) decoder that was a component of a speech-to-speech translation system. The input to our system therefore contained the kinds of recognition errors and disfluencies typically found in ASR output. This experiment serves to determine the robustness of the domain prediction to real-world speech input. The speech recognition process in this experiment had a word accuracy of 88.4% and a sentence accuracy of 67.2% . The results shown in Table 3 clearly demonstrate that the proposed method is able to improve the translation performance, even when speech recognition errors are present in the input sentence.

6.3 Comparison with previous methods

In this section we compare the proposed method to other comtemporary methods: the cluster language model (CLM) and the sentence mixture model (SMix). The experimental results for these methods were reported by RWTH Aachen University in IWSLT06 (Mauser et al., 2006). We evaluated our method using the same training and evaluation corpora. These corpora were used as the training and development corpora in the IWSLT06 Chinese to English open track, the details are given in Table 4. The English side of the training corpus was the same as that used in the earlier Japanese to English experiments reported in this paper. Each sentence in the evaluation corpus had seven reference translations. Our baseline performance was slightly different from that reported in the RWTH experiments (21.9 BLEU socre for RWTH’s system and 21.7 for our system). Therefore, their improved baseline is shown for comparison. The results are shown in Table 5. The improvements over the baseline of our method in both BLEU and NIST (Doddington 2002) score were greater than those for both CLM and SMix. In particular, our method showed improvent in both the BLEU and NIST scores, this is in contrast to the CLM and SMix methods which both degraded the translation performance in terms of the NIST score.

Table 5: Comparison results with previous methods

	BLEU	NIST	WER	PER
RWTH	21.9	6.31	66.4	50.8
Our	21.7	6.79	70.9	51.2
CLM	+0.6	-0.22	-2.7	-1.1
SMix	+0.2	-0.06	-1.1	-0.9
Proposed	+1.1	+0.17	-1.1	-0.5

6.4 Clustering of the monolingual corpus

Finally, we evaluated the proposed method when an additional monolingual corpus was incorporated. For this experiment, we used the Chinese and English bilingual corpora that were used in the NIST MT06 evaluation (NIST 2006). The size of the bilingual training corpus was 2.9M sentence pairs. For the language model training, an additional monolingual corpus of 1.5M English sentences was used. NIST 2006 development (evaluation set for NIST 2005) is used for evaluation. In this experiment, the test set language model perplexity of a model built on only the monolingual corpus was considerably lower than that of a model built from only the target language sentences from the bilingual corpus. Therefore, we would expect the use of this monolingual corpus to be an important factor affecting the quality of the translation system. These perplexities were 299.9 for the model built on only the bilingual corpus, 200.1 for the model built on only the monolingual corpus, and 192.5 for the model built on a combination of the bilingual and monolingual corpora. For the domain specific models, 50 clusters were created from the bilingual and monolingual corpora. In this experiment, only the domain specific language model was used. The experimental results are shown in Table 6. The results in the table show that the incorporation of the additional monolingual data has a pronounced beneficial effect on performance, the performance improved according to all of the evaluation measures.

Table 2: Japanese to English translation evaluation scores

	BLEU	NIST	WER	PER	Meteor	TER
Baseline	52.38	9.316	42.87	33.21	70.63	35.46
Domain Specific LM	53.66	9.349	41.73	32,27	71.39	34.17
Domain Specific TM	54.30	9.333	41.64	32.50	71.77	33.80
Domain Specific LM+TM	55.09	9.451	41.05	31.63	72.09	33.20

Table 3: Evaluation using ASR output

	BLEU	NIST	WER	PER	Meteor	TER
Baseline	48.17	8.892	47.05	36.86	67.40	39.36
Domain Specific LM	48.94	8.900	46.26	36.37	67.98	38.42
Domain Specific TM	49.11	8.842	45.78	36.55	68.01	37.88
Domain Specific LM+TM	50.12	9.001	45.26	35.80	68.05	37.22

7 Conclusion

We have proposed a technique that utilizes domain specific models based on bilingual clustering for statistical machine translation. It is well-known that domain specific modeling can result in better performance. However, in many cases, the target domain is not known or can change dynamically. In such cases, domain determination and domain specific translation must be performed simultaneously during the translation process. In the proposed method, a bilingual corpus was clustered using an entropy reduction based method. The resulting bilingual clusters are regarded as domains. Domain specific language and translation models are created from the data within each bilingual cluster. When a source sentence is to be translated, its domain is first predicted. The domain prediction method selects the cluster that assigns the lowest language model perplexity to the given source sentence. Translation then proceeds using a language model and translation model that are specific to the domain predicted for the source sentence.

In our experiments we used a corpus from the travel domain (the subset of the BTEC corpus that was used in IWSLT06). Our experimental results clearly demonstrate the effectiveness of our method. In the Japanese to English translation experiments, the use of our proposed method improved the BLEU score by 2.7 points (from 52.4 to 55.1). We compared our approach to two previous methods, the cluster language model and sentence mixture model. In our experiments the proposed method yielded higher scores than either of the competitive methods in terms of both BLEU and NIST. Moreover, our method may also be augmented when an additional monolingual corpus is avaliable for building the language model. Using this approach we were able to further improve translation performance on the data from the NIST MT06 evaluation task.

A Sample of Cluster

A.1 Cluster 1

- E: do you do alterations
  J: 直し は し てい ます か (naoshi wa shi tei masu ka)

- E: what’s the newest color in this season
  J: 今年 の 新色 は どれ です か (kotoshi no shinshoku wa dore desu ka)

- E: are there any baseball games today
  J: 今日 野球 の 試合 は あり ます か (kyou yakyu no shiai wa ari masu ka)

- E: where’s the nearest perfumery
  J: 最寄り の 香水 店 は どこ です か (moyori no kousui ten wa doko desu ka)

- E: how much is the breakfast
  J: 朝食 は いくら です か (choshoku wa ikura desu ka)

Table 4: Training and evaluation corpora used for comparison with previous methods

	# of sentence	Total words	Vocabulary size
English Training	40K	315K	9.2K
Chinese Training	40K	304K	18.7K
Chinese Evaluation	489	5,110	1.3K

Table 6: Experimental results with monolingual corpus

	BLEU	NIST	WER	PER	Meteor	TER
Baseline	24.39	7.918	86.51	61.65	53.36	68.21
Proposed	24.95	8.030	85.89	61.27	53.86	67.48

A.2 Cluster 2

- E: mr. aoki yes a single room for two nights
  J: アオキ さん です ね えー シングルルーム で 二 泊 です ね (aoki san desu ne ee shingururumu de 2 haku desu ne)

- E: may i have the key to room two fifteen
  J: 二 一 五 号 室 の 鍵 を 下さい (2 1 5 gou shitsu no kagi o kudasai)

- E: i'd like extension twenty four please
  J: 内線 二 十 四 を 御 願い し ます (naisen 24 o o begai shi masu)

- E: the flight number is se one o three to tokyo on the second of april
  J: フライトナンバー は 東京 行き エスイー 一 ゼロ 三 便 四月 二日 の 便 です (furaitonanba wa tokyo iki s e 1 0 3 bin 4 gatsu futsuka no bin desu)

- E: delta airlines flight one one two boarding is delayed
  J: デルタ航空 一 一 二 便 は 搭乗 が 遅れ てい ます (derutakouku 1 1 2 bin wa tojo ga okure tei masu)

B Sample of Different Translation Results

1. Ref: your room is number two ten
   Base: your room this is room two o one
   LM: your room is this is room two one zero
   TM: your room is room two o one
   LM+TM: your room is this is room two one zero

2. Ref: where is a spot where there are a lot of fish
   Base: i'm spot where is the lot of fish
   LM: where is the spot are a lot of fish
   TM: i'm spot where is the lot of fish
   LM+TM: where is the spot are a lot of fish

3. Ref: i don't like the design
   Base: design i don't like it
   LM: i don't like it design
   TM: i don't like the design
   LM+TM: i don't like the design

4. Ref: where can i contact you
   Base: where contact if i may
   LM: where contact if i can
   TM: where can i contact
   LM+TM: where can i contact

5. Ref: where is a police station where japanese is understood
   Base: japanese where's the police station
   LM: japanese where's the police station
   TM: where's the police station where someone understands japanese
   LM+TM: where's the police station where someone understands japanese

References

K. Seymore, R. Rosenfeld, ”Using Story Topics for Language Model Adaptation,” Proc. EUROSPEECH, pp. 1987-1990, 1997.

David Carter, ”Improving Language Models by Clustering Training Sentences,” Proc. ACL, pp. 59-64, 1994.

S. Hasan, H. Ney, ”Clustered Language Models Based on Regular Expressions for SMT,” Proc. EAMT, Budapest, Hungary, May 2005.

R. M. Iyer and M. Ostendorf, ”Modeling Long Distance Dependence in Language: Topic mixture versus dynamic cache models,” IEEE Transactions on Speech and Audio Processing, 1994.

T. Takezawa, E. Sumita, F. Sugaya, H. Yamamoto, S. Yamamoto, ”Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world,” Proc. Conference on Language Resource and Evaluation, May 2002.

Genichiro Kikui, Eiichiro Sumita, Toshiyuki Takezawa, Seiichi Yamamoto, ”Creating Corpora for Speech-to-Speech Translation,” Proc. EUROSPEECH, pp. 381-384, 2003.

M. Paul, ”Overview of the IWSLT 2006 Evaluation Campaign,” IWSLT 2006, Nov. 2006.

R. Zens, F. J. Och, H. Ney, ”Phrase-based statistical machine translation,” 25th German Conference on Artificial Inteligence, Sep 2002.

F. J. Och, H. Ney, ”A Systematic Comparison of Various Statistical Alignment Models,” Computational Linguistics, No. 1, Vol. 29, pp. 19-51, 2003.

S. M. Katz, ”Estimation of Probabilities from Sparse Data for Language Model Component of a Speech Recognizer,” IEEE Trans. on Acoustics, Speech, and Signal Processing, pp. 400-401, 1987.

A. Stolcke, ”SRILM - An Extensible Language Model Toolkit,” http://www.speech.sri.com/projects/srilm/

P. Koehn, ”PHARAOH: A beam search decoder for phrase-based statistical machine translation models,” http://www.isi.edu/publications/licensed-sw/pharaoh/

F. J. Och, ”Minimum error rate training for statistical machine trainslation,” Proc. ACL, 2003.

K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, ”Bleu: a method for automatic evaluation of machine translation,” Proc. ACL, 2002.

A. Mauser, R. Zens, E. Matusov, S. Hasan, H. Ney, ”The RWTH Statistical Machine Translation System for IWSLT 2006 Evaluation,” IWSLT 2006, Nov. 2006.

G. Doddington, ”Automatic evaluation of machine translation quality using n-gram co-occurrence statistics,” Proc. ARPA Workshop on Human Language Technology, 2002.

NIST 2006 Machine Translation Evaluation, http://www.nist.gov/speech/tests/mt/ mt06eval_official_results.html

A Systematic Comparison of Training Criteria for Statistical Machine Translation

Richard Zens and Saša Hasan and Hermann Ney

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6 – Computer Science Department
RWTH Aachen University, D-52056 Aachen, Germany
{zens,hasan,ney}@cs.rwth-aachen.de

Abstract

We address the problem of training the free parameters of a statistical machine translation system. We show significant improvements over a state-of-the-art minimum error rate training baseline on a large Chinese-English translation task. We present novel training criteria based on maximum likelihood estimation and expected loss computation. Additionally, we compare the maximum a-posteriori decision rule and the minimum Bayes risk decision rule. We show that, not only from a theoretical point of view but also in terms of translation quality, the minimum Bayes risk decision rule is preferable.

1 Introduction

Once we specified the Bayes decision rule for statistical machine translation, we have to address three problems (Ney, 2001):

- the search problem, i.e. how to find the best translation candidate among all possible target language sentences;
- the modeling problem, i.e. how to structure the dependencies of source and target language sentences;
- the training problem, i.e. how to estimate the free parameters of the models from the training data.

Here, the main focus is on the training problem. We will compare a variety of training criteria for statistical machine translation. In particular, we are considering criteria for the log-linear parameters or model scaling factors. We will introduce new training criteria based on maximum likelihood estimation and expected loss computation. We will show that some achieve significantly better results than the standard minimum error rate training of (Och, 2003).

Additionally, we will compare two decision rules, the common maximum a-posteriori (MAP) decision rule and the minimum Bayes risk (MBR) decision rule (Kumar and Byrne, 2004). We will show that the minimum Bayes risk decision rule results in better translation quality than the maximum a-posteriori decision rule for several training criteria.

The remaining part of this paper is structured as follows: first, we will describe related work in Sec. 2. Then, we will briefly review the baseline system, Bayes decision rule for statistical machine translation and automatic evaluation metrics for machine translation in Sec. 3 and Sec. 4, respectively. The novel training criteria are described in Sec. 5 and Sec. 6. Experimental results are reported in Sec. 7 and conclusions are given in Sec. 8.

2 Related Work

The most common modeling approach in statistical machine translation is to use a log-linear combination of several sub-models (Och and Ney, 2002). In (Och and Ney, 2002), the log-linear weights were tuned to maximize the mutual information criterion (MMI). The current state-of-the-art is to optimize these parameters with respect to the final evaluation criterion; this is the so-called minimum error rate training (Och, 2003).

Minimum Bayes risk decoding for machine trans-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 524–532, Prague, June 2007. ©2007 Association for Computational Linguistics

lation was introduced in (Kumar and Byrne, 2004). It was shown that MBR outperforms MAP decoding for different evaluation criteria. Further experiments using MBR for Bleu were performed in (Venugopal et al., 2005; Ehling et al., 2007). Here, we will present additional evidence that MBR decoding is preferable over MAP decoding.

Tillmann and Zhang (2006) describe a perceptron style *algorithm* for training millions of features. Here, we focus on the comparison of different training *criteria*.

Shen et al. (2004) compared different algorithms for tuning the log-linear weights in a reranking framework and achieved results comparable to the standard minimum error rate training.

An annealed minimum risk approach is presented in (Smith and Eisner, 2006) which outperforms both maximum likelihood and minimum error rate training. The parameters are estimated iteratively using an annealing technique that minimizes the risk of an expected-BLEU approximation, which is similar to the one presented in this paper.

3 Baseline System

In statistical machine translation, we are given a source language sentence $f_1^J = f_1 \dots f_j \dots f_J$, which is to be translated into a target language sentence $e_1^I = e_1 \dots e_i \dots e_I$. Statistical decision theory tells us that among all possible target language sentences, we should choose the sentence which minimizes the expected loss, also called Bayes risk:

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmin}_{I,e_1^I} \left\{ \sum_{I',e'^{I'}_1} Pr(e'^{I'}_1|f_1^J) \cdot L(e_1^I, e'^{I'}_1) \right\}$$

Here, $L(e_1^I, e'^{I'}_1)$ denotes the loss function under consideration. It measures the loss (or errors) of a candidate translation e_1^I assuming the correct translation is $e'^{I'}_1$. In the following, we will call this decision rule the MBR rule (Kumar and Byrne, 2004). This decision rule is optimal in the sense that any other decision rule will result (on average) in at least as many errors as the MBR rule. Despite this, most SMT systems do *not* use the MBR decision rule. The most common approach is to use the maximum a-posteriori (MAP) decision rule. Thus, we select the hypothesis which maximizes the posterior probability $Pr(e_1^I|f_1^J)$:

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I,e_1^I} \left\{ Pr(e_1^I|f_1^J) \right\}$$

This is equivalent to the MBR decision rule under a 0-1 loss function:

$$L_{0-1}(e_1^I, e'^{I'}_1) = \begin{cases} 0 & \text{if } e_1^I = e'^{I'}_1 \\ 1 & \text{else} \end{cases}$$

Hence, the MAP decision rule is optimal for the sentence or string error rate. It is *not* necessarily optimal for other evaluation metrics such as the Bleu score. One reason for the popularity of the MAP decision rule might be that, compared to the MBR rule, its computation is simpler.

The posterior probability $Pr(e_1^I|f_1^J)$ is modeled directly using a log-linear combination of several models (Och and Ney, 2002):

$$p_{\lambda_1^M}(e_1^I|f_1^J) = \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J)\right)}{\sum\limits_{I',e'^{I'}_1} \exp\left(\sum_{m=1}^M \lambda_m h_m(e'^{I'}_1, f_1^J)\right)} \tag{1}$$

This approach is a generalization of the source-channel approach (Brown et al., 1990). It has the advantage that additional models $h(\cdot)$ can be easily integrated into the overall system.

The denominator represents a normalization factor that depends only on the source sentence f_1^J. Therefore, we can omit it in case of the MAP decision rule during the search process and obtain:

$$\hat{e}_1^{\hat{I}} \quad = \quad \operatorname*{argmax}_{I,e_1^I} \left\{ \sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J) \right\}$$

Note that the denominator affects the results of the MBR decision rule and, thus, cannot be omitted in that case.

We use a state-of-the-art phrase-based translation system similar to (Koehn, 2004; Mauser et al., 2006) including the following models: an n-gram language model, a phrase translation model and a word-based lexicon model. The latter two models are used for both directions: $p(f|e)$ and $p(e|f)$. Additionally, we use a word penalty, phrase penalty and a distortion penalty.

In the following, we will discuss the so-called training problem (Ney, 2001): how do we train the free parameters λ_1^M of the model? The current state-of-the-art is to use minimum error rate training (MERT) as described in (Och, 2003). The free parameters are tuned to directly optimize the evaluation criterion.

Except for the MERT, the training criteria that we will consider are additive at the sentence-level. Thus, the training problem for a development set with S sentences can be formalized as:

$$\hat{\lambda}_1^M = \operatorname*{argmax}_{\lambda_1^M} \sum_{s=1}^{S} F(\lambda_1^M, (e_1^I, f_1^J)_s) \qquad (2)$$

Here, $F(\cdot, \cdot)$ denotes the training criterion that we would like to maximize and $(e_1^I, f_1^J)_s$ denotes a sentence pair in the development set. The optimization is done using the Downhill Simplex algorithm from the Numerical Recipes book (Press et al., 2002). This is a general purpose optimization procedure with the advantage that it does not require the derivative information. Before we will describe the details of the different training criteria in Sec. 5 and 6, we will discuss evaluation metrics in the following section.

4 Evaluation Metrics

The automatic evaluation of machine translation is currently an active research area. There exists a variety of different metrics, e.g., word error rate, position-independent word error rate, BLEU score (Papineni et al., 2002), NIST score (Doddington, 2002), METEOR (Banerjee and Lavie, 2005), GTM (Turian et al., 2003). Each of them has advantages and shortcomings.

A popular metric for evaluating machine translation quality is the Bleu score (Papineni et al., 2002). It has certain shortcomings for comparing different machine translation systems, especially if comparing conceptually different systems, e.g. phrase-based versus rule-based systems, as shown in (Callison-Burch et al., 2006). On the other hand, Callison-Burch concluded that the Bleu score is reliable for comparing variants of the same machine translation system. As this is exactly what we will need in our experiments and as Bleu is currently the most popular metric, we have chosen it as our primary evaluation metric. Nevertheless, most of the methods we will present can be easily adapted to other automatic evaluation metrics.

In the following, we will briefly review the computation of the Bleu score as some of the training criteria are motivated by this. The Bleu score is a combination of the geometric mean of n-gram precisions and a brevity penalty for too short translation hypotheses. The Bleu score for a translation hypothesis e_1^I and a reference translation $\hat{e}_1^{\hat{I}}$ is computed as:

$$\text{Bleu}(e_1^I, \hat{e}_1^{\hat{I}}) = \text{BP}(I, \hat{I}) \cdot \prod_{n=1}^{4} \text{Prec}_n(e_1^I, \hat{e}_1^{\hat{I}})^{1/4}$$

with

$$\text{BP}(I, \hat{I}) = \begin{cases} 1 & \text{if } I \geq \hat{I} \\ \exp\left(1 - I/\hat{I}\right) & \text{if } I < \hat{I} \end{cases}$$

$$\text{Prec}_n(e_1^I, \hat{e}_1^{\hat{I}}) = \frac{\sum\limits_{w_1^n} \min\{C(w_1^n|e_1^I), C(w_1^n|\hat{e}_1^{\hat{I}})\}}{\sum\limits_{w_1^n} C(w_1^n|e_1^I)} \qquad (3)$$

Here, $C(w_1^n|e_1^I)$ denotes the number of occurrences of an n-gram w_1^n in a sentence e_1^I. The denominators of the n-gram precisions evaluate to the number of n-grams in the hypothesis, i.e. $I - n + 1$.

The n-gram counts for the Bleu score computation are usually collected over a whole document. For our purposes, a sentence-level computation is preferable. A problem with the sentence-level Bleu score is that the score is zero if not at least one four-gram matches. As we would like to avoid this problem, we use the smoothed sentence-level Bleu score as suggested in (Lin and Och, 2004). Thus, we increase the nominator and denominator of $\text{Prec}_n(\cdot, \cdot)$ by one for $n > 1$. Note that we will use the sentence-level Bleu score only during training. The evaluation on the development and test sets will be carried out using the standard Bleu score, i.e. at the corpus level. As the MERT baseline does not require the use of the sentence-level Bleu score, we use the standard Bleu score for training the baseline system.

In the following, we will describe several criteria for training the log-linear parameters λ_1^M of our model. For notational convenience, we assume that there is just one reference translation. Nevertheless, the methods can be easily adapted to the case of multiple references.

5 Maximum Likelihood

5.1 Sentence-Level Computation

A popular approach for training parameters is maximum likelihood estimation (MLE). Here, the goal is to maximize the joint likelihood of the parameters and the training data. For log-linear models, this results in a nice optimization criterion which is convex and has a single optimum. It is equivalent to the maximum mutual information (MMI) criterion. We obtain the following training criterion:

$$F_{ML-S}(\lambda_1^M, (e_1^I, f_1^J)) = \log p_{\lambda_1^M}(e_1^I|f_1^J)$$

A problem that we often face in practice is that the correct translation might not be among the candidates that our MT system produces. Therefore, (Och and Ney, 2002; Och, 2003) defined the translation candidate with the minimum word-error rate as pseudo reference translation. This has some bias towards minimizing the word-error rate. Here, we will use the translation candidate with the maximum Bleu score as pseudo reference to bias the system towards the Bleu score. However, as pointed out in (Och, 2003), there is no reason to believe that the resulting parameters are *optimal* with respect to translation quality measured with the Bleu score.

The goal of this sentence-level criterion is to discriminate the single correct translation against all the other ”incorrect” translations. This is problematic as, even for human experts, it is very hard to define a single best translation of a sentence. Furthermore, the alternative target language sentences are not all equally bad translations. Some of them might be very close to the correct translation or even equivalent whereas other sentences may have a completely different meaning. The sentence-level MLE criterion does not distinguish these cases and is therefore a rather harsh training criterion.

5.2 N-gram Level Computation

As an alternative to the sentence-level MLE, we performed experiments with an n-gram level MLE. Here, we limit the order of the n-grams and assume conditional independence among the n-gram probabilities. We define the log-likelihood (LLH) of a target language sentence e_1^I given a source language sentence f_1^J as:

$$F_{ML-N}(\lambda_1^M, (e_1^I, f_1^J)) = \sum_{n=1}^{N} \sum_{w_1^n \in e_1^I} \log p_{\lambda_1^M}(w_1^n|f_1^J)$$

Here, we use the n-gram posterior probability $p_{\lambda_1^M}(w_1^n|f_1^J)$ as defined in (Zens and Ney, 2006). The n-gram posterior distribution is smoothed using a uniform distribution over all possible n-grams.

$$\begin{aligned} p_{\lambda_1^M}(w_1^n|f_1^J) &= \alpha \cdot \frac{N_{\lambda_1^M}(w_1^n, f_1^J)}{\sum_{w'^n_1} N_{\lambda_1^M}(w'^n_1, f_1^J)} \\ &\quad + (1-\alpha) \cdot \frac{1}{V^n} \end{aligned}$$

Here, V denotes the vocabulary size of the target language; thus, V^n is the number of possible n-grams in the target language. We define $N_{\lambda_1^M}(w_1^n, f_1^J)$ as in (Zens and Ney, 2006):

$$N_{\lambda_1^M}(w_1^n, f_1^J) = \sum_{I, e_1^I} \sum_{i=1}^{I-n+1} p_{\lambda_1^M}(e_1^I|f_1^J) \cdot \delta(e_i^{i+n-1}, w_1^n) \quad (4)$$

The sum over the target language sentences is limited to an N-best list, i.e. the N best translation candidates according to the baseline model. In this equation, we use the Kronecker function $\delta(\cdot, \cdot)$, i.e. the term $\delta(e_i^{i+n-1}, w_1^n)$ evaluates to one if and only if the n-gram w_1^n occurs in the target sentence e_1^I starting at position i.

An advantage of the n-gram level computation of the likelihood is that we do not have to define pseudo-references as for the sentence-level MLE. We can easily compute the likelihood for the human reference translation. Furthermore, this criterion has the desirable property that it takes partial correctness into account, i.e. it is not as harsh as the sentence-level criterion.

6 Expected Bleu Score

According to statistical decision theory, one should maximize the expected gain (or equivalently minimize the expected loss). For machine translation, this means that we should optimize the expected Bleu score, or any other preferred evaluation metric.

6.1 Sentence-Level Computation

The expected Bleu score for a given source sentence f_1^J and a reference translation $\hat{e}_1^{\hat{I}}$ is defined as:

$$\mathbb{E}[\text{Bleu}|\hat{e}_1^{\hat{I}}, f_1^J] = \sum_{e_1^I} Pr(e_1^I|f_1^J) \cdot \text{Bleu}(e_1^I, \hat{e}_1^{\hat{I}})$$

Here, $Pr(e_1^I|f_1^J)$ denotes the true probability distribution over the possible translations e_1^I of the given source sentence f_1^J. As this probability distribution is unknown, we approximate it using the log-linear translation model $p_{\lambda_1^M}(e_1^I|f_1^J)$ from Eq. 1. Furthermore, the computation of the expected Bleu score involves a sum over all possible translations e_1^I. This sum is approximated using an N-best list, i.e. the N best translation hypotheses of the MT system. Thus, the training criterion for the sentence-level expected Bleu computation is:

$$F_{EB-S}(\lambda_1^M, (\hat{e}_1^{\hat{I}}, f_1^J)) = \sum_{e_1^I} p_{\lambda_1^M}(e_1^I|f_1^J) \cdot \text{Bleu}(e_1^I, \hat{e}_1^{\hat{I}})$$

An advantage of the sentence-level computation is that it is straightforward to plug in alternative evaluation metrics instead of the Bleu score. Note that the minimum error rate training (Och, 2003) uses only the target sentence with the *maximum* posterior probability whereas, here, the whole probability *distribution* is taken into account.

6.2 N-gram Level Computation

In this section, we describe a more fine grained computation of the expected Bleu score by exploiting its particular structure. Hence, this derivation is specific for the Bleu score but should be easily adaptable to other n-gram based metrics. We can rewrite the expected Bleu score as:

$$\begin{aligned}\mathbb{E}[\text{Bleu}|\hat{e}_1^{\hat{I}}, f_1^J] &= \mathbb{E}[\text{BP}|\hat{I}, f_1^J] \\ &\cdot \prod_{n=1}^{4} \mathbb{E}[\text{Prec}_n|\hat{e}_1^{\hat{I}}, f_1^J]^{1/4}\end{aligned}$$

We assumed conditional independence between the brevity penalty BP and the n-gram precisions Prec_n. Note that although these independence assumptions do not hold, the resulting parameters might work well for translation. In fact, we will show that this criterion is among the best performing ones in Sec. 7. This type of independence assumption is typical within the naive Bayes classifier framework. The resulting training criterion that we will use in Eq. 2 is then:

$$\begin{aligned}F_{EB-N}(\lambda_1^M, (\hat{e}_1^{\hat{I}}, f_1^J)) &= \mathbb{E}_{\lambda_1^M}[\text{BP}|\hat{I}, f_1^J] \\ &\cdot \prod_{n=1}^{4} \mathbb{E}_{\lambda_1^M}[\text{Prec}_n|\hat{e}_1^{\hat{I}}, f_1^J]^{1/4}\end{aligned}$$

We still have to define the estimators for the expected brevity penalty as well as the expected n-gram precision:

$$\mathbb{E}_{\lambda_1^M}[\text{BP}|\hat{I}, f_1^J] = \sum_I \text{BP}(I, \hat{I}) \cdot p_{\lambda_1^M}(I|f_1^J)$$

$$\mathbb{E}_{\lambda_1^M}[\text{Prec}_n|\hat{e}_1^{\hat{I}}, f_1^J] = \qquad (5)$$

$$\frac{\sum_{w_1^n} p_{\lambda_1^M}(w_1^n|f_1^J) \sum_c \min\{c, C(w_1^n|\hat{e}_1^{\hat{I}})\} \cdot p_{\lambda_1^M}(c|w_1^n, f_1^J)}{\sum_{w_1^n} p_{\lambda_1^M}(w_1^n|f_1^J) \sum_c c \cdot p_{\lambda_1^M}(c|w_1^n, f_1^J)}$$

Here, we use the sentence length posterior probability $p_{\lambda_1^M}(I|f_1^J)$ as defined in (Zens and Ney, 2006) and the n-gram posterior probability $p_{\lambda_1^M}(w_1^n|f_1^J)$ as described in Sec. 5.2. Additionally, we predict the number of occurrences c of an n-gram. This information is necessary for the so-called clipping in the Bleu score computation, i.e. the $\min$ operator in the nominator of formulae Eq. 3 and Eq. 5. The denominator of Eq. 5 is the expected number of n-grams in the target sentence, whereas the nominator denotes the expected number of correct n-grams.

To predict the number of occurrences within a translation hypothesis, we use relative frequencies smoothed with a Poisson distribution. The mean of the Poisson distribution $\mu(w_1^n, f_1^J, \lambda_1^M)$ is chosen to be the mean of the unsmoothed distribution.

$$\begin{aligned}p_{\lambda_1^M}(c|w_1^n, f_1^J) &= \beta \cdot \frac{N_{\lambda_1^M}(c, w_1^n, f_1^J)}{N_{\lambda_1^M}(w_1^n, f_1^J)} \\ &+ (1-\beta) \cdot \frac{\mu(w_1^n, f_1^J, \lambda_1^M)^c \cdot e^{-c}}{c!}\end{aligned}$$

Table 1: Chinese-English TC-Star task: corpus statistics.

			Chinese	English
Train		Sentence pairs	8.3 M	
		Running words	197 M	238 M
		Vocabulary size	224 K	389 K
Dev		Sentences	1 019	2 038
		Running words	26 K	51 K
Eval	2006	Sentences	1 232	2 464
		Running words	30 K	62 K
	2007	Sentences	917	1 834
		Running words	21 K	45 K

with

$$\mu(w_1^n, f_1^J, \lambda_1^M) = \sum_c c \cdot \frac{N_{\lambda_1^M}(c, w_1^n, f_1^J)}{N_{\lambda_1^M}(w_1^n, f_1^J)}$$

Note that in case the mean $\mu(w_1^n, f_1^J, \lambda_1^M)$ is zero, we do not need the distribution $p_{\lambda_1^M}(c|w_1^n, f_1^J)$. The smoothing parameters α and β are both set to 0.9.

7 Experimental Results

7.1 Task Description

We perform translation experiments on the Chinese-English TC-Star task. This is a broadcast news speech translation task used within the European Union project TC-Star[1]. The bilingual training data consists of virtually all publicly available LDC Chinese-English corpora. The 6-gram language model was trained on the English part of the bilingual training data and additional monolingual English parts from the GigaWord corpus. We use the modified Kneser-Ney discounting as implemented in the SRILM toolkit (Stolcke, 2002).

Annual public evaluations are carried out for this task within the TC-Star project. We will report results on manual transcriptions, i.e. the so-called verbatim condition, of the official evaluation test sets of the years 2006 and 2007. There are two reference translations available for the development and test sets. The corpus statistics are shown in Table 1.

7.2 Translation Results

In Table 2, we present the translation results for different training criteria for the development set and the two blind test sets. The reported case-sensitive Bleu scores are computed using the `mteval-v11b.pl`[2] tool using two reference translations, i.e. BLEUr2n4c. Note that already the baseline system (MERT-Bleu) would have achieved the first rank in the official TC-Star evaluation 2006; the best Bleu score in that evaluation was 16.1%.

The MBR hypotheses were generated using the algorithm described in (Ehling et al., 2007) on a 10 000-best list.

On the development data, the MERT-Bleu achieves the highest Bleu score. This seems reasonable as it is the objective of this training criterion.

The maximum likelihood (MLE) criteria perform somewhat worse under MAP decoding. Interestingly, the MBR decoding can compensate this to a large extent: all criteria achieve a Bleu score of about 18.9% on the development set. The benefits of MBR decoding become even more evident on the two test sets. Here, the MAP results for the sentence-level MLE criterion are rather poor compared to the MERT-Bleu. Nevertheless, using MBR decoding results in very similar Bleu scores for most of the criteria on these two test sets. We can therefore support the claim of (Smith and Eisner, 2006) that MBR tends to have better generalization capabilities.

The n-gram level MLE criterion seems to perform better than the sentence-level MLE criterion, especially on the test sets. The reasons might be that there is no need for the use of pseudo references as described in Sec. 5 and that partial correctness is taken into account.

The best results are achieved using the expected Bleu score criteria described in Sec. 6. Here, the sentence level and n-gram level variants achieve more or less the same results. The overall improvement on the Eval'06 set is about 1.0% Bleu absolute for MAP decoding and 0.9% for MBR decoding. On the Eval'07 set, the improvements are even larger, about 1.8% Bleu absolute for MAP and 1.1% Bleu for MBR. All these improvements are statistically significant at the 99% level using a pairwise significance test[3].

Given that currently the most popular approach is to use MERT-Bleu MAP decoding, the overall im-

[1] http://www.tc-star.org

[2] http://www.nist.gov/speech/tests/mt/resources/scoring.htm

[3] The tool for computing the significance test was kindly provided by the National Research Council Canada.

Table 2: Translation results: Bleu scores [%] for the Chinese-English TC-Star task for various training criteria (MERT: minimum error rate training; MLE: maximum likelihood estimation; $\mathbb{E}[\text{Bleu}]$: expected Bleu score) and the maximum a-posteriori (MAP) as well as the minimum Bayes risk (MBR) decision rule.

			Development		Eval'06		Eval'07	
Decision Rule			MAP	MBR	MAP	MBR	MAP	MBR
Training Criterion	MERT-Bleu (baseline)		**19.5**	**19.4**	16.7	17.2	22.2	23.0
	MLE	sentence-level	17.8	18.9	14.8	17.1	18.9	22.7
		n-gram level	18.6	18.8	17.0	17.8	22.8	23.5
	$\mathbb{E}[\text{Bleu}]$	sentence-level	19.1	18.9	17.5	**18.1**	23.5	**24.1**
		n-gram level	18.6	18.8	**17.7**	17.6	**24.0**	**24.0**

provement is about 1.4% absolute for the Eval'06 set and 1.9% absolute on the Eval'07 set.

Note that the MBR decision rule almost always outperforms the MAP decision rule. In the rare cases where the MAP decision rule yields better results, the difference in terms of Bleu score are small and *not* statistically significant.

We also investigated the effect of the maximum n-gram order for the n-gram level maximum likelihood estimation (MLE). The results are shown in Figure 1. We observe an increase of the Bleu score with increasing maximum n-gram order for the development corpus. On the evaluation sets, however, the maximum is achieved if the maximum n-gram order is limited to four. This seems intuitive as the Bleu score uses n-grams up to length four. However, one should be careful here: the differences are rather small, so it might be just statistical noise.

Some translation examples from the Eval'07 test set are shown in Table 3 for different training criteria under the maximum a-posteriori decision rule.

8 Conclusions

We have presented a systematic comparison of several criteria for training the log-linear parameters of a statistical machine translation system. Additionally, we have compared the maximum a-posteriori with the minimum Bayes risk decision rule.

We can conclude that the expected Bleu score is not only a theoretically sound training criterion, but also achieves the best results in terms of Bleu score. The improvement over a state-of-the-art MERT baseline is 1.3% Bleu absolute for the MAP decision rule and 1.1% Bleu absolute for the MBR decision rule for the large Chinese-English TC-Star speech translation task.

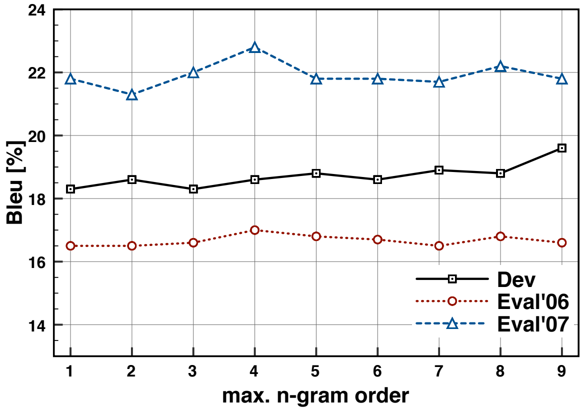


Figure 1: Effect of the maximum n-gram order on the Bleu score for the n-gram level maximum likelihood estimation under the maximum a-posteriori decision rule.

We presented two methods for computing the expected Bleu score: a sentence-level and an n-gram level approach. Both yield similar results. We think that the n-gram level computation has certain advantages: The n-gram posterior probabilities could be computed from a word graph which would result in more reliable estimates. Whether this pays off in terms of translation quality is left open for future work.

Another interesting result of our experiments is that the MBR decision rule seems to be less affected by sub-optimal parameter settings.

Although it is well-known that the MBR decision rule is more appropriate than the MAP decision rule, the latter is more popular in the SMT community (and many other areas of natural language processing). Our results show that it can be beneficial to

Table 3: Translation examples from the Eval'07 test set for different training criteria and the maximum a-posteriori decision rule. (MERT: minimum error rate training, MLE-S: sentence-level maximum likelihood estimation, $\mathbb{E}$[Bleu]: sentence-level expected Bleu)

Criterion	Translation
Reference 1	Saving Private Ryan ranks the third on the box office revenue list which is also a movie that is possible to win an 1999 Oscar award
2	Saving Private Ryan ranked third in the box office income is likely to compete in the nineteen ninety-nine Oscar Awards
MERT-Bleu	Saving private Ryan in box office income is possible ranked third in 1999 Oscar a film
MLE-S	Saving private Ryan box office revenue ranked third is possible in 1999 Oscar a film
$\mathbb{E}$[Bleu]-S	Saving private Ryan ranked third in the box office income is also likely to run for the 1999 Academy Awards a film
Reference 1	The following problem is whether people in countries like China and Japan and other countries will choose Euros rather than US dollars in international business activities in the future
2	The next question is whether China or Japan or other countries will choose to use Euros instead of US dollars when they conduct international business in the future
MERT-Bleu	The next question is in China or Japan international business activities in the future they will not use the Euro dollar
MLE-S	The next question was either in China or Japan international business activities in the future they will adopt the Euro instead of the dollar
$\mathbb{E}$[Bleu]-S	The next question was in China or Japan in the international business activities in the future they will adopt the Euro instead of the US dollar
Reference 1	The Chairman of the European Commission Jacques Santer pointed out in this September that the financial crisis that happened in Russia has not affected people's confidence in adopting the Euro
2	European Commission President Jacques Santer pointed out in September this year that Russia's financial crisis did not shake people's confidence for planning the use of the Euro
MERT-Bleu	President of the European Commission Jacques Santer on September this year that the Russian financial crisis has not shaken people 's confidence in the introduction of the Euro
MLE-S	President of the European Commission Jacques Santer September that the Russian financial crisis has not affected people 's confidence in the introduction of the Euro
$\mathbb{E}$[Bleu]-S	President of the European Commission Jacques Santer pointed out that Russia 's financial crisis last September has not shaken people 's confidence in the introduction of the Euro
Reference 1	After many years of friction between Dutch and French speaking Belgians all of them now hope to emphasize their European identities
2	After years of friction between Belgium's Dutch-speaking and French-speaking people they now all wish to emphasize their European identity
MERT-Bleu	Belgium's Dutch-speaking and French-speaking after many years of civil strife emphasized that they now hope that Europeans
MLE-S	Belgium's Dutch-speaking and francophone after years of civil strife that they now hope that Europeans
$\mathbb{E}$[Bleu]-S	Belgium's Dutch-speaking and French-speaking after many years of civil strife it is now want to emphasize their European identity

use the MBR decision rule. On the other hand, the computation of the MBR hypotheses is more time consuming. Therefore, it would be desirable to have a more efficient algorithm for computing the MBR hypotheses.

Acknowledgments

This material is partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023, and was partly funded by the European Union under the integrated project TC-STAR (Technology and Corpora for Speech to Speech Translation, IST-2002-FP6-506738, http://www.tc-star.org).

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proc. *Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 65–72, Ann Arbor, MI, June.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, June.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In Proc. *11th Conf. of the Europ. Chapter of the Assoc. for Computational Linguistics (EACL)*, pages 249–256, Trento, Italy, April.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In Proc. *ARPA Workshop on Human Language Technology*.

Nicola Ehling, Richard Zens, and Hermann Ney. 2007. Minimum Bayes risk decoding for BLEU. In Proc. *45th Annual Meeting of the Assoc. for Computational Linguistics (ACL): Poster Session*, Prague, Czech Republic, June.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In Proc. *6th Conf. of the Assoc. for Machine Translation in the Americas (AMTA)*, pages 115–124, Washington DC, September/October.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pages 169–176, Boston, MA, May.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In Proc. *COLING '04: The 20th Int. Conf. on Computational Linguistics*, pages 501–507, Geneva, Switzerland, August.

Arne Mauser, Richard Zens, Evgeny Matusov, Saša Hasan, and Hermann Ney. 2006. The RWTH statistical machine translation system for the IWSLT 2006 evaluation. In Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pages 103–110, Kyoto, Japan, November.

Hermann Ney. 2001. Stochastic modelling: from pattern classification to language translation. In Proc. *39th Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Data-Driven Machine Translation*, pages 1–5, Morristown, NJ, July.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In Proc. *40th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, July.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In Proc. *41st Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proc. *40th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, July.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2002. *Numerical Recipes in C++*. Cambridge University Press, Cambridge, UK.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pages 177–184, Boston, MA, May.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL): Poster Session*, pages 787–794, Sydney, Australia, July.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In Proc. *Int. Conf. on Speech and Language Processing (ICSLP)*, volume 2, pages 901–904, Denver, CO, September.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical MT. In Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pages 721–728, Sydney, Australia, July.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. 2003. Evaluation of machine translation and its evaluation. Technical Report Proteus technical report 03-005, Computer Science Department, New York University.

Ashish Venugopal, Andreas Zollmann, and Alex Waibel. 2005. Training and evaluating error minimization rules for statistical machine translation. In Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 208–215, Ann Arbor, MI, June.

Richard Zens and Hermann Ney. 2006. N-gram posterior probabilities for statistical machine translation. In Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Proc. Workshop on Statistical Machine Translation*, pages 72–77, New York City, NY, June.

Phrase Reordering Model Integrating Syntactic Knowledge for SMT

Dongdong Zhang, Mu Li, Chi-Ho Li, Ming Zhou

Microsoft Research Asia
Beijing, China
{dozhang,muli,chl,mingzhou}@microsoft.com

Abstract

Reordering model is important for the statistical machine translation (SMT). Current phrase-based SMT technologies are good at capturing local reordering but not global reordering. This paper introduces syntactic knowledge to improve global reordering capability of SMT system. Syntactic knowledge such as boundary words, POS information and dependencies is used to guide phrase reordering. Not only constraints in syntax tree are proposed to avoid the reordering errors, but also the modification of syntax tree is made to strengthen the capability of capturing phrase reordering. Furthermore, the combination of parse trees can compensate for the reordering errors caused by single parse tree. Finally, experimental results show that the performance of our system is superior to that of the state-of-the-art phrase-based SMT system.

1 Introduction

In the last decade, statistical machine translation (SMT) has been widely studied and achieved good translation results. Two kinds of SMT system have been developed, one is phrase-based SMT and the other is syntax-based SMT.

In phrase-based SMT systems (Koehn et al., 2003; Koehn, 2004), foreign sentences are firstly segmented into *phrases* which consists of adjacent words. Then source phrases are translated into target phrases respectively according to knowledge usually learned from bilingual parallel corpus. Finally the most likely target sentence based on a certain statistical model is inferred by combining and reordering the target phrases with the aid of search algorithm. On the other hand, syntax-based SMT systems (Liu et al., 2006; Yamada et al., 2001) mainly depend on parse trees to complete the translation of source sentence.

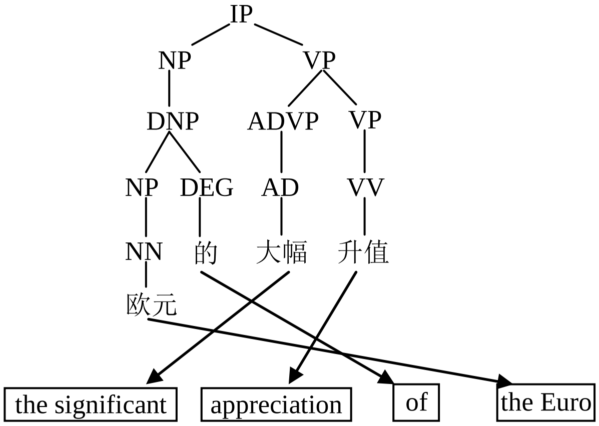


Figure 1: A reordering example

As studied in previous SMT projects, language model, translation model and reordering model are the three major components in current SMT systems. Due to the difference between the source and target languages, the order of target phrases in the target sentence may differ from the order of source phrases in the source sentence. To make the translation results be closer to the target language style, a mathematic model based on the statistic theory is constructed to reorder the target phrases. This statistic model is called as *reordering model*. As shown in Figure 1, the order of the translations of “欧元” and “的” is changed. The order of the

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 533–540, Prague, June 2007. ©2007 Association for Computational Linguistics

translation of “欧元/的” and “大幅/升值” is altered as well. The former reordering case with the smaller distance is usually referred as *local reordering* and the latter with the longer distance reordering as *global reordering*. Phrase-based SMT system can effectively capture the local word reordering information which is common enough to be observed in training data. But it is hard to model global phrase reordering. Although syntactic knowledge used in syntax-based SMT systems can help reorder phrases, the resulting model is usually much more complicated than a phrase-based system.

There have been considerable amount of efforts to improve the reordering model in SMT systems, ranging from the fundamental distance-based distortion model (Och and Ney, 2004; Koehn et al., 2003), flat reordering model (Wu, 1996; Zens et al., 2004; Kumar et al., 2005), to lexicalized reordering model (Tillmann, 2004; Kumar et al., 2005; Koehn et al., 2005), hierarchical phrase-based model (Chiang, 2005), and maximum entropy-based phrase reordering model (Xiong et al., 2006). Due to the absence of syntactic knowledge in these systems, the ability to capture global reordering knowledge is not powerful. Although syntax-based SMT systems (Yamada et al., 2001; Quirk et al., 2005; Liu et al., 2006) are good at modeling global reordering, their performance is subject to parsing errors to a large extent.

In this paper, we propose a new method to improve reordering model by introducing syntactic information. Syntactic knowledge such as boundary of sub-trees, part-of-speech (POS) and dependency relation is incorporated into the SMT system to strengthen the ability to handle global phrase reordering. Our method is different from previous syntax-based SMT systems in which the translation process was modeled based on specific syntactic structures, either phrase structures or dependency relations. In our system, syntactic knowledge is used just to decide where we should combine adjacent phrases and what their reordering probability is. For example, according to the syntactic information in Figure 1, the phrase translation combination should take place between “大幅” and “升值” rather than between “的” and “大幅”. Moreover, the non-monotone phrase reordering should occur between “欧元/的” and “大幅/升值” rather than between “欧元/的” and “大幅”. We train a maximum entropy model, which is able to integrate rich syntactic knowledge, to estimate phrase reordering probabilities. To enhance the performance of phrase reordering model, some modification on the syntax trees are also made to relax the phrase reordering constraints. Additionally, the combination of other kinds of syntax trees is introduced to overcome the deficiency of single parse tree. The experimental results show that the performance of our system is superior to that of the state-of-art phrase-based SMT system.

The roadmap of this paper is: Section 2 gives the related work. Section 3 introduces our model. Section 4 explains the generalization of reordering knowledge. The procedures of training and decoding are described in Section 5 and Section 6 respectively. The experimental results are shown in Section 7. Section 8 concludes the paper.

2 Related Work

The Pharaoh system (Koehn et al., 2004) is well known as the typical phrase-based SMT system. Its reordering model is designed to penalize translation according to jump distance regardless of linguistic knowledge. This method just works well for language pairs that trend to have similar word-orders and it has nothing to do with global reordering.

A straightforward reordering model used in (Wu, 1996; Zens et al., 2004; Kumar et al., 2005) is to assign constant probabilities to monotone reordering and non-monotone reordering, which can be flexible depending on the different language pairs. This method is also adopted in our system for non-peer phrase reordering.

The lexicalized reordering model was studied in (Tillmann, 2004; Kumar et al., 2005; Koehn et al., 2005). Their work made a step forward in integrating linguistic knowledge to capture reordering. But their methods have the serious data sparseness problem.

Beyond standard phrase-based SMT system, a CKY style decoder was developed in (Xiong et al., 2006). Their method investigated the reordering of any two adjacent phrases. The limited linguistic knowledge on the boundary words of phrases is used to construct the phrase reordering model. The basic difference to our method is that no syntactic knowledge is introduced to guide the global phrase reordering in their system. Besides boundary words, our phrase reordering model also integrates more significant syntactic knowledge such as POS information and dependencies from the syntax tree, which can avoid some intractable phrase reordering errors.

A hierarchical phrase-based model was proposed by (Chiang, 2005). In his method, a synchronous CFG is used to reorganize the phrases into hierarchical ones and grammar rules are automatically learned from corpus. Different from his work, foreign syntactic knowledge is introduced into the synchronous grammar rules in our method to restrict the arbitrary phrase reordering.

Syntax-based SMT systems (Yamada et al., 2001; Quirk et al., 2005; Liu et al., 2006) totally depend on syntax structures to complete phrase translation. They can capture global reordering by simply swapping the children nodes of a parse tree. However, there are also reordering cases which do not agree with syntactic structures. Furthermore, their model is usually much more complex than a phrase-based system. Our method exactly attempts to integrate the advantages of phrase-based SMT system and syntax-based SMT system to improve the phrase reordering model. Phrase translation in our system is independent of syntactic structures.

3 The Model

In our work, we focus on building a better reordering model with the help of source parsing information. Although we borrow some fundamental elements from a phrase-based SMT system such as the use of bilingual phrases as basic translation unit, we are more interested in introducing syntactic knowledge to strengthen the ability to handle global reordering phenomena in translation.

3.1 Definitions

Given a foreign sentence *f* and its syntactic parse tree *T*, each leaf in *T* corresponds to a single word in *f* and each sub-tree of *T* exactly covers a phrase f_i in *f* which is called as *linguistic phrase*. Except linguistic phrases, any other phrase is regarded as *non-linguistic phrase*. The *height* of phrase f_i is defined as the distance between the root node of *T* and the root node of the maximum sub-tree which exactly covers f_i. For example, in Figure 1 the phrase "大幅" has the maximum sub-tree rooting at ADJP and its height is 3. The height of phrase "的" is 4 since its maximum sub-tree roots at ADBP instead of AD. If two adjacent phrases have the same height, we regard them as *peer phrases*.

In our model, we make use of *bilingual phrases* as well, which refer to source-target aligned phrase pairs extracted using the same criterion as most phrase-based systems (Och and Ney, 2004).

3.2 Model

Similar to the work in Chiang (2005), our translation model can be formulated as a weighted synchronous context free grammar derivation process. Let *D* be a derivation that generates a bilingual sentence pair ⟨*f*, *e*⟩, in which *f* is the given source sentence, the statistical model that is used to predict the translation probability *p*(*e*|*f*) is defined over *Ds* as follows:

$$p(e|f) \propto p(D) \propto p_{lm}(e)^{\lambda_{lm}} \times \prod_{i} \prod_{X \to \langle \gamma, \alpha \rangle \in D} \phi_i(X \to \langle \gamma, \alpha \rangle)^{\lambda_i}$$

where $p_{lm}(e)$ is the language model, $\Phi_i(X \rightarrow \langle \gamma, \alpha \rangle)$ is a feature function defined over the derivation rule $X \rightarrow \langle \gamma, \alpha \rangle$, and λ_i is its weight.

Although theoretically it is ideal for translation reorder modeling by constructing a synchronous context free grammar based on bilingual linguistic parsing trees, it is generally a very difficult task in practice. In this work we propose to use a small synchronous grammar constructed on the basis of bilingual phrases to model translation reorder probability and constraints by referring to the source syntactic parse trees. In the grammar, the source / target words serve as terminals, and the bilingual phrases and combination of bilingual phrases are presented with non-terminals. There are two non-terminals in the grammar except the start symbol *S*: *Y* and *Z*. The general derivation rules are defined as follows:

a) Derivations from non-terminal to non-terminals are restricted to binary branching forms;
b) Any non-terminals that derives a list of terminals, or any combination of two non-terminals, if the resulting source string won't cause any cross-bracketing problems in the source parse tree (it exactly corresponds to a linguistic phrase in binary parse trees), are reduced to *Y*;
c) Otherwise, they are reduced to *Z*.

Table 1 shows a complete list of derivation rules in our synchronous context grammar. The first nine grammar rules are used to constrain phrase reordering during phrase combination. The last two rules are used to represent bilingual phrases. Rule (10) is the start grammar rule to generate the entire sentence translation.

Rule Name	Rule Content
Rule (1)	$Y\rightarrow\langle Y_1Y_2, Y_1Y_2\rangle$
Rule (2)	$Y\rightarrow\langle Y_1Y_2, Y_2Y_1\rangle$
Rule (3)	$Y\rightarrow\langle Z_1Z_2, Z_1Z_2\rangle$
Rule (4)	$Y\rightarrow\langle Y_1Z_2,Y_1Z_2\rangle$
Rule (5)	$Y\rightarrow\langle Z_1Y_2, Z_1Y_2\rangle$
Rule (6)	$Z\rightarrow\langle Y_1Z_2, Y_1Z_2\rangle$
Rule (7)	$Z\rightarrow\langle Z_1Y_2, Z_1Y_2\rangle$
Rule (8)	$Z\rightarrow\langle Z_1Z_2, Z_1Z_2\rangle$
Rule (9)	$Z\rightarrow\langle Y_1Y_2, Y_1Y_2\rangle$
Rule (10)	$S\rightarrow\langle Y_1,Y_1\rangle$
Rule (11)	$\mathrm{Z}\rightarrow\langle Z_1, Z_1\rangle$
Rule (12)	$Y\rightarrow\langle Y_1,Y_1\rangle$

Table 1: Synchronous grammar rules

Rule (1) and Rule (2) are only applied to two adjacent peer phrases. Note that, according to the constraints of foreign syntactic structures, only Rule (2) among all rules in Table 1 can be applied to conduct non-monotone phrase reordering in our framework. This can avoid arbitrary phrase reordering. For example, as shown in Figure 1, Rule (1) is applied to the monotone combination of phrases “欧元” and “的”, and Rule (2) is applied to the non-monotone combination of phrases “欧元/的” and “大幅/升值”. However, the non-monotone combination of “的” and “大幅” is not allowed in our method since there is no proper rule for it.

Non-linguistic phrases are involved in Rule (3)~(9). We do not allow these grammar rules for non-monotone combination of non-peer phrases, which really harm the translation results as proved in experimental results. Although these rules violate the syntactic constraints, they not only provide the option to leverage non-linguistic translation knowledge to avoid syntactic errors but also take advantage of phrase local reordering capabilities. Rule (3) and Rule (8) are applied to the combination of two adjacent non-linguistic phrases. Rule (4)~(7) deal with the situation where one is a linguistic phrase and the other is a non-linguistic phrase. Rule (9) is applied to the combination of two adjacent linguistic phrases but their combination result is not a linguistic phrase.

Rule (11) and Rule (12) are applied to generate bilingual phrases learned from training corpus.

Table 2 demonstrates an example how these rules are applied to translate the foreign sentence “欧元/的/大幅/升值” into the English sentence “the significant appreciation of the Euro”.

Step	Partial derivations	Rule
1	$\mathrm{S}\rightarrow\langle Y_1, Y_1\rangle$	(10)
2	$\rightarrow\langle Y_2Y_3, Y_3Y_2\rangle$	(2)
3	$\rightarrow\langle Y_4Y_5Y_3, Y_3Y_5Y_4\rangle$	(2)
4	$\rightarrow\langle$欧元 Y_5Y_3, Y_3Y_5 the Euro$\rangle$	(12)
5	$\rightarrow\langle$欧元 的 Y_3, Y_3 of the Euro$\rangle$	(12)
6	$\rightarrow\langle$欧元 的 Y_6Y_7, Y_6Y_7 of the Euro$\rangle$	(1)
7	$\rightarrow\langle$欧元 的 大幅 Y_7, the significant Y_7 of the Euro$\rangle$	(12)
8	$\rightarrow\langle$欧元 的 大幅 升值, the significant appreciation of the Euro$\rangle$	(12)

Table 2: Example of application for rules

However, there are always other kinds of bilingual phrases extracted directly from training corpus, such as ⟨欧元, the Euro⟩ and ⟨的 大幅 升值, ’s significant appreciation⟩, which can produce different candidate sentence translations. Here, the phrase “的 大幅 升值” is a non-linguistic phrase. The above derivations can also be rewritten as $S\rightarrow\langle Y_1, Y_1\rangle\rightarrow\langle Y_2Z_3,Y_2Z_3\rangle\rightarrow\langle$ 欧 元 Z_3, the Euro $Z_3\rangle\rightarrow\langle$欧元的 大幅 升值, the Euro ’s significant appreciation$\rangle$, where Rule (10), (4), (12) and (11) are applied respectively.

3.3 Features

Similar to the default features in Pharaoh (Koehn, Och and Marcu 2003), we used following features to estimate the weight of our grammar rules. Note that different rules may have different features in our model.

- The lexical weights $p_{lex}(\gamma|\alpha)$ and $p_{lex}(\alpha|\gamma)$ estimating how well the words in α translate the words in γ. This feature is only applicable to Rule (11) and Rule (12).
- The phrase translation weights $p_{phr}(\gamma|\alpha)$ and $p_{phr}(\alpha|\gamma)$ estimating how well the terminal words of α translate the terminal words of γ, This feature is only applicable to Rule (11) and Rule (12).
- A word penalty $\exp(|\alpha|)$, where $|\alpha|$ denotes the count of terminal words of α. This feature is only applicable to Rule (11) and Rule (12).
- A penalty exp(1) for grammar rules analogous to Pharaoh's penalty which allows the model to learn a preference for longer or shorter derivations. This feature is applicable to all rules in Table 1.
- Score for applying the current rule. This feature is applicable to all rules in Table 1. We will explain the score estimation in detail in Section 3.4.

3.4 Scoring of Rules

Based on the syntax constraints and involved non-terminal types, we separate the grammar rules into three groups to estimate their application scores which are also treated as reordering probabilities.

For Rule (1) and Rule (2), they strictly comply with the syntactic structures. Given two peer phrases, we have two choices to use one of them. Thus, we use maximum entropy (ME) model algorithm to estimate their reordering probabilities separately, where the boundary words of foreign phrases and candidate target translation phrases, POS information and dependencies are integrated as features. As listed in Table 3, there are totally twelve categories of features used to train the ME model. In fact, the probability of Rule (1) is just equal to the supplementary probability of Rule (2), and vice versa.

For Rule (3)~(9), according to the syntactic structures, their application is determined since there is only one choice to complete reordering, which is similar to the "glue rules" in Chiang (2005). Due to the appearance of non-linguistic phrases, non-monotone phrase reordering is not allowed in these rules. We just assign these rules a constant score trained using our implementation of Minimum Error Rate Training (Och, 2003b), which is 0.7 in our system.

For Rule (10)~(12), they are also determined rules since there is no other optional rules competing with them. Constant score is simply assigned to them as well, which is 1.0 in our system.

Fea.	Description
LS1	First word of first foreign phrase
LS2	First word of second foreign phrase
RS1	Last word of first foreign phrase
RS2	Last word of second foreign phrase
LT1	First word of first target phrase
LT2	First word of second target phrase
RT1	Last word of first target phrase
RT2	Last word of second target phrase
LPos	POS of the node covering first foreign phrase
RPos	POS of the node covering second foreign phrase
Cpos	POS of the node covering the combination of foreign phrases
DP	Dependency between the nodes covering two single foreign phrases respectively

Table 3: Feature categories used for ME model

4 The Generalization of Reordering Knowledge

4.1 Enriching Parse Trees

The grammar rules proposed in Section 3 are only applied to binary syntax tree nodes. For *n*-ary syntax trees (*n*>2), some modification is needed to generate more peer phrases. As shown in Figure 2(a), the syntactic tree of Chinese sentence "广东省/高新技术/产品/出口" (Guangdong/high-tech/products/export), parsed by the Stanford Parser (Klein, 2003), has a 3-ary sub-tree. Referring to its English translation result "export of high-tech products in Guangdong", we understand there should be a non-monotone combination between the phrases "广东省" and "高新技术/产品". However, "高新技术/产品" is not a linguistic phrase though its component phrases “高新技术” and “产品” are peer phrases. To avoid the conflict with the Rule (2), we just add some extra *virtual nodes* in the *n*-ary sub-trees to make sure that only binary sub-trees survive in the modified parse tree. Figure 2(b) is the modification result of the syntactic tree from Figure 2(a), where two virtual nodes with the new distinguishable POS of M are added.

In general, we add virtual nodes for each set of the continuous peer phrases and let them have the same height. Thus, for a *n*-ary sub-tree, there are $\sum_{i=1}^{n-1}(n-i) = (n-1)^2/2$ virtual nodes being added where $n>2$. The phrases exactly covered by the virtual nodes are called as *virtual peer phrases*.

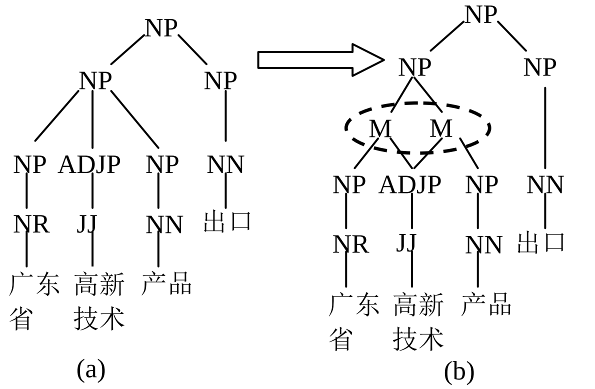


Figure 2: Example of syntax tree modification

4.2 Combination of Parse Trees

It is well known that parse errors in syntactic trees always are inescapable even if the state-of-the-art parser is used. Incorrect syntactic knowledge may harm the reordering probability estimation. To minimize the impact of parse error of a single tree, more parse trees are introduced. To support the combination of parse trees, the synchronous grammar rules are applied independently, but they will compete against each other with the effect of other models such as language model.

In our system, we combine the parse trees generated respectively by Stanford parser (Klein, 2003) and a dependency parser developed by (Zhou, 2000). Compared with the Stanford parser, the dependency parser only conducts shallow syntactic analysis. It is powerful to identify the base NPs and base VPs and their dependencies. Additionally, dependency parser runs much faster. For example, it took about three minutes for the dependency parser to parse one thousand sentences with average length of 25 words, but the Stanford parser needs about one hour to complete the same work. More importantly, as shown in the experimental results, the dependency parser can achieve the comparable quality of final translation results with Stanford parser in our system.

5 The Decoder

We developed a CKY style decoder to complete the sentence translation. A two-dimension array *CA* is constructed to store all the local candidate phrase translation and each valid cell CA_{ij} in *CA* corresponds to a foreign phrase where *i* is the phrase start position and *j* is the phrase end position. The cells in *CA* are filled in a bottom-up way. Firstly we fill in smaller cells with the translation in bilingual phrases learned from corpus. Then the candidate translation in the larger cell CA_{ij} is generated based on the content in smaller adjacent cells CA_{ik} and CA_{k+1j} with the *monotone combination* and *non-monotone combination*, where $i \le k \le j$. To reduce the cost of system resources, the well known pruning methods, such as histogram pruning, threshold pruning and recombination, are used to only keep the top *N* candidate translation in each cell.

6 Training

Similar to most state-of-the-art phrase-based SMT systems, we use the SRI toolkit (Stolcke, 2002) for language model training and Giza++ toolkit (Och and Ney, 2003) for word alignment. For reordering model training, two kinds of parse trees for each foreign sentence in the training corpus were obtained through the Stanford parser (Klein, 2003) and a dependency parser (Zhou, 2000). After that, we picked all the foreign linguistic phrases of the same sentence according to syntactic structures. Based on the word alignment results, if the aligned target words of any two adjacent foreign linguistic phrases can also be formed into two valid adjacent phrase according to constraints proposed in the phrase extraction algorithm by Och (2003a), they will be extracted as a reordering training sample. Finally, the ME modeling toolkit developed by Zhang (2004) is used to train the reordering model over the extracted samples.

7 Experimental Results and Analysis

We conducted our experiments on Chinese-to-English translation task of NIST MT-05 on a 3.0GHz system with 4G RAM memory. The bilingual training data comes from the FBIS corpus. The Xinhua news in GIGAWORD corpus is used to train a four-gram language model. The development set used in our system is the NIST MT-02 evaluation test data.

For phrase extraction, we limit the maximum length of foreign and English phrases to 3 and 5 respectively. But there is no phrase length constraint for reordering sample extraction. About 1.93M and 1.1M reordering samples are extracted from the FBIS corpus based on the Stanford parser and the dependency parser respectively. To reduce the search space in decoder, we set the histogram pruning threshold to 20 and relative pruning threshold to 0.1.

In the following experiments, we compared our system performance with that of the other state-of-the-art systems. Additionally, the effect of some strategies on system performance is investigated as well. Case-sensitive BLEU-4 score is adopted to evaluate system performance.

7.1 Comparing with Baseline SMT system

Our baseline system is Pharaoh (Koehn, 2004). Xiong's system (Xiong, et al., 2006) which used ME model to train the reordering model is also regarded as a competitor. To have a fair comparison, we used the same language model and translation model for these three systems. The experimental results are showed in Table 4.

System	Bleu Score
Pharaoh	0.2487
Xiong's System	0.2616
Our System	0.2737

Table 4: Performance against baseline system

These three systems are the same in that the final sentence translation results are generated by the combination of local phrase translation. Thus, they are capable of local reordering but not global reordering. The phrase reordering in Pharaoh depends only on distance distortion information which does not contain any linguistic knowledge. The experimental result shows that the performance of both Xiong's system and our system is better than that of Pharaoh. It proves that linguistic knowledge can help the global reordering probability estimation. Additionally, our system is superior to Xiong's system in which only use phrase boundary words to guide global reordering. It indicates that syntactic knowledge is more powerful to guide global reordering than boundary words. On the other hand, it proves the importance of syntactic knowledge constraints in avoiding the arbitrary phrase reordering.

7.2 Syntactic Error Analysis

Rule (3)~(9) in Section 3 not only play the role to compensate for syntactic errors, but also take the advantage of the capability of capturing local phrase reordering. However, the non-monotone combination for non-peer phrases is really harmful to system performance. To prove these ideas, we conducted experiments with different constrains.

Constraints	Bleu Score
All rules in Table 1 used	0.2737
Allowing the non-monotone combination of non-peer phrases	0.2647
Rule (3)~(9) are prohibited	0.2591

Table 5: About non-peer phrase combination

From the experimental results shown in Table 5, just as claimed in other previous work, the combination between non-linguistic phrases is useful and cannot be abandoned. On the other hand, if we relax the constraint of non-peer phrase combination (that is, allowing non-monotone combination for on-peer phrases), some more serious errors in non-syntactic knowledge is introduced, thereby degrading performance from 0.2737 to 0.2647.

7.3 Effect of Virtual Peer Phrases

As discussed in Section 4, for *n*-ary nodes (n>2) in the original syntax trees, the relationship among *n*-ary sub-trees is always not clearly captured. To give them the chance of free reordering, we add the virtual peer nodes to make sure that the combination of a set of peer phrases can still be a peer phrase. An experiment was done to compare with the case where the virtual peer nodes were not added to *n*-ary syntax trees. The Bleu score dropped to 26.20 from 27.37, which shows the virtual nodes have great effect on system performance.

7.4 Effect of Mixed Syntax Trees

In this section, we conducted three experiments to investigate the effect of constituency parse tree and dependency parse tree. Over the same platform, we tried to use only one of them to complete the translation task. The experimental results are shown in Table 6.

Surprisingly, there is no significant difference in performance. The reason may be that both parsers produce approximately equivalent parse results. However, the combination of syntax trees outperforms merely only one syntax tree. This suggests that the N-best syntax parse trees may enhance the quality of reordering model.

Situation	**Bleu Score**
Dependency parser only	0.2667
Stanford parser only	0.2670
Mixed parsing trees	0.2737

Table 6: Different parsing tree

8 Conclusion and Future Work

In this paper, syntactic knowledge is introduced to capture global reordering of SMT system. This method can not only inherit the advantage of local reordering ability of standard phrase-based SMT system, but also capture the global reordering as the syntax-based SMT system. The experimental results showed the effectiveness of our method.

In the future work, we plan to improve the reordering model by introducing N-best syntax trees and exploiting richer syntactic knowledge.

References

David Chiang. 2005. *A hierarchical phrase-based model for statistical machine translation*. In *Proceedings of ACL 2005*.

Franz Josef Och. 2003a. *Statistical Machine Translation: From Single-Word Models to Alignment Templates Thesis*.

Franz Josef Och. 2003b. *Minmum Error Rate Training in Statistical Machine Translation*. In *Proceedings for ACL 2003*.

Franz Josef Och, Hermann Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Models*. Computational Linguistics, 29:19-51.

Franz Josef Och and Hermann Ney. 2004. *The alignment template approach to statistical machine translation*. Computational Linguistics, 30:417−449.

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. In *Proceedings of ACL 2003*.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. *Statistical Phrase-Based Translation.* In *Proceedings of HLT/NAACL 2003*.

Philipp Koehn. 2004. *Pharaoh: a Beam Search Decoder for Phrased-Based Statistical Machine Translation Models.* In *Proceedings of AMTA 2004*.

Shankar Kumar and William Byrne. 2005. *Local phrase reordering models for statistical machine translation*. In *Proceedings of HLT-EMNLP 2005*.

Yang Liu, Qun Liu, Shouxun Lin. 2006. *Tree-to-String Alignment Template for Statistical Machine Translation*. In *Proceedings of COLING-ACL 2006*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. *Dependency treelet translation: Syntactically informed phrasal SMT*. In *Proceedings of ACL 2005*.

Andreas Stolcke. 2002. *SRILM-An Extensible Language Modeling Toolkit*. In *Proceedings of ICSLP 2002*.

Christoph Tillmann. 2004. *A block orientation model for statistical machine translation*. In *Proceedings of HLT-NAACL 2004*.

Dekai Wu. 1996. *A Polynomial-Time Algorithm for Statistical Machine Translation*. In *Proceedings of ACL 1996*.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. *Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation*. In *Proceedings of COLING-ACL 2006*.

Kenji Yamada and Kevin Knight. 2001. *A syntax based statistical translation model*. In *Proceedings of ACL 2001*.

Le Zhang. 2004. Maximum Entropy Modeling Toolkit for Python and C++. Available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. *Reordering Constraints for Phrase-Based Statistical Machine Translation.* In *Proceedings of CoLing 2004*.

Ming Zhou. 2000. *A block-based robust dependency parser for unrestricted Chinese text*. The second Chinese Language Processing Workshop attached to ACL2000.

Identification and Resolution of Chinese Zero Pronouns: A Machine Learning Approach

Shanheng Zhao and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
3 Science Drive 2
Singapore 117543
{zhaosh, nght}@comp.nus.edu.sg

Abstract

In this paper, we present a machine learning approach to the identification and resolution of Chinese anaphoric zero pronouns. We perform both identification and resolution automatically, with two sets of easily computable features. Experimental results show that our proposed learning approach achieves anaphoric zero pronoun resolution accuracy comparable to a previous state-of-the-art, heuristic rule-based approach. To our knowledge, our work is the first to perform both identification and resolution of Chinese anaphoric zero pronouns using a machine learning approach.

1 Introduction

Coreference resolution is the task of determining whether two or more noun phrases refer to the same entity in a text. It is an important task in discourse analysis, and successful coreference resolution benefits many natural language processing applications such as information extraction, question answering, etc.

In the literature, much of the work on coreference resolution is for English text (Soon et al., 2001; Ng and Cardie, 2002b; Yang et al., 2003; McCallum and Wellner, 2005). Publicly available corpora for coreference resolution are mostly in English, e.g., the Message Understanding Conference tasks (MUC6 and MUC7)[1]. Relatively less work has been done on coreference resolution for Chinese. Recently, the ACE Entity Detection and Tracking (EDT) task[2] included annotated Chinese corpora for coreference resolution. Florian et al. (2004) and Zhou et al. (2005) reported research on Chinese coreference resolution.

A prominent phenomenon in Chinese coreference resolution is the prevalence of zero pronouns. A zero pronoun (ZP) is a gap in a sentence which refers to an entity that supplies the necessary information for interpreting the gap. An anaphoric zero pronoun (AZP) is a zero pronoun that corefers to one or more overt noun phrases present in the preceding text. Zero pronouns occur much more frequently in Chinese compared to English, and pose a unique challenge in coreference resolution for Chinese. For example, Kim (2000) conducted a study to compare the use of overt subjects in English, Chinese, and other languages. He found that the use of overt subjects in English is over 96%, while this percentage is only 64% for Chinese, indicating that zero pronouns (lack of overt subjects) are much more prevalent in Chinese.

Chinese zero pronouns have been studied in linguistics research (Li and Thompson, 1979; Li, 2004), but only a small body of prior work in computational linguistics deals with Chinese zero pronoun identification and resolution (Yeh and Chen, 2004; Converse, 2006). To our knowledge, all previous research on zero pronoun identification and resolution in Chinese uses hand-engineered rules or heuristics, and our present work is the first to perform both identification and resolution of Chinese

[1] http://www-nlpir.nist.gov/related_projects/muc/

[2] http://www.nist.gov/speech/tests/ace/

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 541–550, Prague, June 2007. ©2007 Association for Computational Linguistics

anaphoric zero pronouns using a machine learning approach.

The rest of this paper is organized as follows. In Section 2, we give the task definition, and describe the corpus used in our evaluation and the evaluation metrics. We then give an overview of our approach in Section 3. Anaphoric zero pronoun identification and resolution are presented in Section 4 and 5, respectively. We present the experimental results in Section 6 and related work in Section 7, and conclude in Section 8.

2 Task Definition

2.1 Zero Pronouns

As mentioned in the introduction, a zero pronoun (ZP) is a gap in a sentence which refers to an entity that supplies the necessary information for interpreting the gap. A coreferential zero pronoun is a zero pronoun that corefers to one or more overt noun phrases present in the same text.

Just like a coreferential noun phrase, a coreferential zero pronoun can also corefer to a noun phrase in the preceding or following text, called anaphoric or cataphoric, respectively. Most coreferential zero pronouns in Chinese are anaphoric. In the corpus used in our evaluation, 98% of the coreferential zero pronouns have antecedents. Hence, for simplicity, we only consider anaphoric zero pronouns (AZP) in this work. That is, we only attempt to resolve a coreferential zero pronoun to noun phrases preceding it.

Here is an example of an anaphoric zero pronoun from the Penn Chinese TreeBank (CTB) (Xue et al., 2005) (sentence ID=300):

[中国	机电	产品	进出口	
[China	electronic	products	import and export	
贸易]$_1$	继续	增加	，	ϕ_2
trade]$_1$	continues	increasing	,	ϕ_2

占	总	进出口	的
represents	total	import and export	’s
比重	继续	上升	。
ratio	continues	increasing	.

The anaphoric zero pronoun ϕ_2 is coreferring to noun phrase 1. The corresponding parse tree is shown in Figure 1. In CTB, IP refers to a simple clause that does not have complementizers. CP, on the other hand, refers to a clause introduced by a complementizer.

Resolving an anaphoric zero pronoun to its correct antecedent in Chinese is a difficult task. Although gender and number information is available for an overt pronoun and has proven to be useful in pronoun resolution in prior research, a zero pronoun in Chinese, unlike an overt pronoun, provides no such gender or number information. At the same time, identifying zero pronouns in Chinese is also a difficult task. There are only a few overt pronouns in English, Chinese, and many other languages, and state-of-the-art part-of-speech taggers can successfully recognize most of these overt pronouns. However, zero pronouns in Chinese, which are not explicitly marked in a text, are hard to be identified. Furthermore, even if a gap is a zero pronoun, it may not be coreferential. All these difficulties make the identification and resolution of anaphoric zero pronouns in Chinese a challenging task.

2.2 Corpus

We use an annotated third-person pronoun and zero pronoun coreference corpus from Converse (2006)[3]. The corpus contains 205 texts from CTB 3.0, with annotations done directly on the parse trees. In the corpus, coreferential zero pronouns, third-person pronouns, and noun phrases are annotated as coreference chains. If a noun phrase is not in any coreference chain, it is not annotated. If a coreference chain does not contain any third-person pronoun or zero pronoun, the whole chain is not annotated.

A zero pronoun is not always coreferential with some noun phrases. In the corpus, if a zero pronoun is not coreferential with any overt noun phrases, it is assigned one of the following six categories: discourse deictic (#DD), existential (#EXT), inferrable (#INFR), ambiguity between possible referents in the text (#AMB), arbitrary reference (#ARB), and unknown (#UNK). For example, in the following sentence, ϕ_3 refers to an event in the preceding text, with no corresponding antecedent noun phrase. So no antecedent is annotated, and ϕ_3 is labeled as #DD.

香港	著名	财团	长江
Hong Kong	famous	syndicate	Cheung Kong

[3] The data set we obtained is a subset of the one used in Converse (2006).

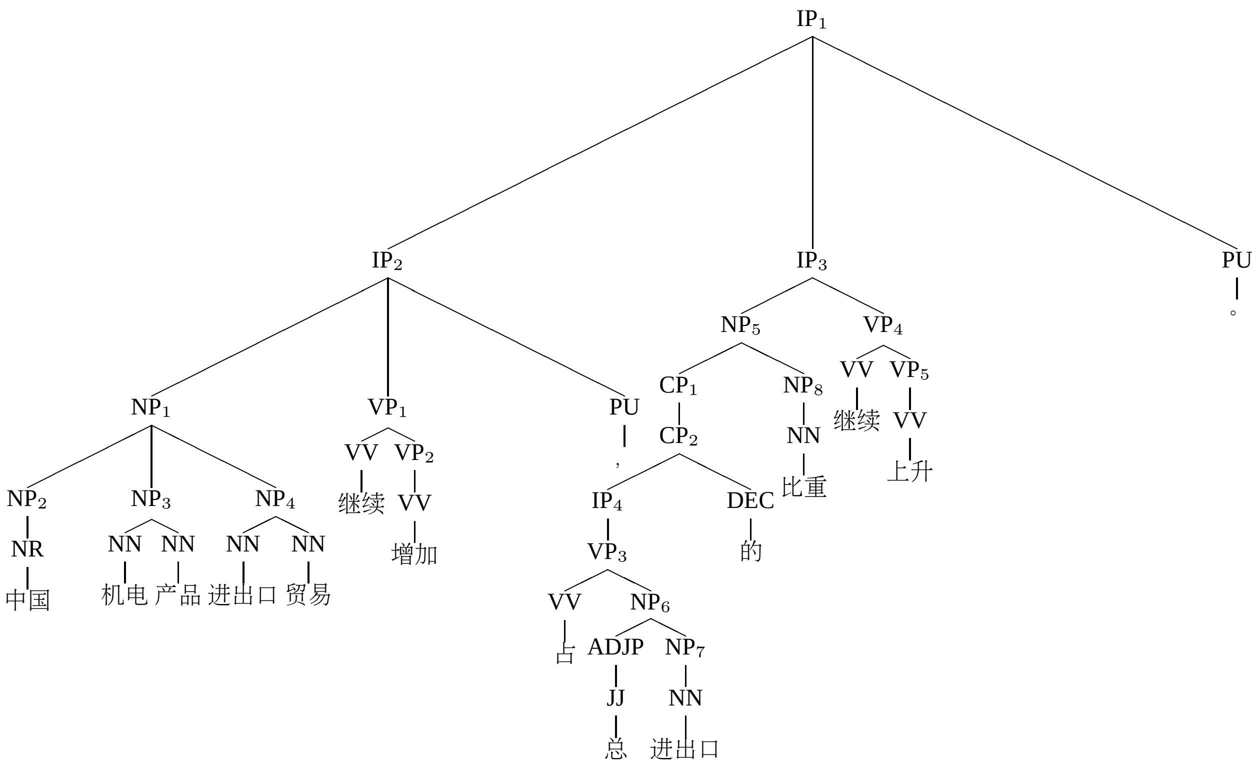


Figure 1: The parse tree which corresponds to the anaphoric zero pronoun example in Section 2.1.

实业 、 百富勤 作为 战略性
Holdings , Peregrine as strategic
投资者 已 购入 了
investors already purchased LE
“ 深业 控股 ” 百分之十二
“ Shenye Holdings ” twenty percent
的 股权 ， ϕ_3 充分 反映 出
’s share , ϕ_3 fully reflects out
投资者 的 信心 。
investors ’s confidence .

Converse (2006) assumed that all correctly identified AZPs and the gold standard parse trees are given as input to her system. She applied the Hobbs algorithm (Hobbs, 1978) to resolve antecedents for the given AZPs.

In our case, we are only interested in zero pronouns with explicit noun phrase referents. If a coreference chain does not contain AZPs, we discard the chain. We also discard the 6 occurrences of zero pronouns with split antecedents, i.e., a zero pronoun with an antecedent that is split into two separate noun phrases. A total of 383 AZPs remain in the data set used in our experiments.

Among the 205 texts in the data set, texts 1–155 are reserved for training, while the remaining texts (156–205) are used for blind test. The statistics of the data set are shown in Table 1.

	Training	Test
Doc ID	1–155	156–205
# Docs	155	50
# Characters	96,338	15,710
# Words	55,348	9,183
# ZPs	665	87
# AZPs	343	40

Table 1: Statistics of training and test data sets.

2.3 Evaluation Metrics

As in previous work on pronoun resolution, we evaluate the accuracy in terms of recall, precision, and F-measure. The overall recall and precision on the test set are computed by micro-averaging over all test instances. The overall F-measure is then computed.

For AZP identification, recall and precision are defined as:

$$Recall_{AZP} = \frac{\text{\# AZP Hit}}{\text{\# AZP in Key}}$$

$$Precision_{AZP} = \frac{\text{\# AZP Hit}}{\text{\# AZP in Response}}$$

An “AZP Hit” occurs when an AZP as reported in the response (system output) has a counterpart in the same position in the gold standard answer key.

For AZP resolution, recall and precision are defined as:

$$Recall_{Resol} = \frac{\text{\# Resol Hit}}{\text{\# AZP in Key}}$$

$$Precision_{Resol} = \frac{\text{\# Resol Hit}}{\text{\# AZP in Response}}$$

A “Resol Hit” occurs when an AZP is correctly identified, and it is correctly resolved to a noun phrase that is in the same coreference chain as provided in the answer key.

3 Overview of Our Approach

In this section, we give an overview of our approach for Chinese AZP identification and resolution.

Typically, the input raw texts need to be processed by a Chinese word segmenter, a part-of-speech (POS) tagger, and a parser sequentially. Although our approach can apply directly to machine-generated parse trees from raw text, in order to minimize errors introduced by preprocessing, and focus mainly on Chinese zero pronoun resolution, we use the gold standard word segmentation, POS tags, and parse trees provided by CTB. However, we remove all null categories and functional tags from the CTB gold standard parse trees. Figure 1 shows a parse tree after such removal.

A set of zero pronoun candidates and a set of noun phrase candidates are then extracted. If W is the leftmost word in the word sequence that is spanned by some VP node, the gap G that is immediately to the left of W qualifies as a ZP candidate. For example, in Figure 1, gaps immediately to the left of the two occurrences of 继续, and 增加, 占, 上升 are all ZP candidates. All noun phrases[4] that are either maximal NPs or modifier NPs qualify as NP candidates. For example, in Figure 1, NP$_1$, NP$_2$, NP$_3$, NP$_5$, and NP$_6$ are all NP candidates. With these ZP and NP candidate extractions, the recalls of ZPs and NPs are 100% and 98.6%, respectively.

After the ZP and NP candidates are determined, we perform AZP identification and resolution in a sequential manner. We build two classifiers, the AZP identification classifier and the AZP resolution classifier. The AZP identification classifier determines the position of AZPs, while the AZP resolution classifier finds an antecedent noun phrase for each AZP identified by the AZP identification classifier. Both classifiers are built using machine learning techniques. The features of both classifiers are largely syntactic features based on parse trees and are easily computed.

We perform 5-fold cross validation on the training data set to tune parameters and to pick the best model. We then retrain the best model with all data in the training data set, and apply it to the blind test set. In the following sections, all accuracies reported on the training data set are based on 5-fold cross validation.

4 Anaphoric Zero Pronoun Identification

We use machine learning techniques to build the AZP identification classifier. The features are described in Table 2.

In the feature description, Z is the ZP candidate. Let W_l and W_r be the words immediately to the left and to the right of Z, respectively, P the parse tree node that is the lowest common ancestor node of W_l and W_r, P_l and P_r the child nodes of P that are ancestor nodes of W_l and W_r, respectively. If Z is the first gap of the sentence, W_l, P, P_l, and P_r are all NA. Furthermore, let V be the highest VP node in the parse tree that is immediately to the right of Z, i.e., the leftmost word in the word sequence that is spanned by V is W_r. If Z is not the first gap in the sentence, define the ceiling node C to be P, otherwise to be the root node of the parse tree. In the example shown in Figure 1, for the ZP candidate ϕ_2 (which is immediately to the left of 占), W_l, W_r, P, P_l, P_r, V, and C are “，”, 占, IP$_1$, IP$_2$, IP$_3$, VP$_3$, and IP$_1$, respectively. Its feature values are also shown in Table 2.

To train an AZP identification classifier, we gen-

[4] A noun phrase can either be NP or QP in CTB. We simply use NP hereafter.

Feature	Description	ϕ_2
First_Gap	If Z is the first gap in the sentence, T; else F.	F
P_l_Is_NP	If Z is the first gap in the sentence, NA; otherwise, if P_l is an NP node, T; else F.	F
P_r_Is_VP	If Z is the first gap in the sentence, NA; otherwise, if P_r is a VP node, T; else F.	F
P_l_Is_NP & P_r_Is_VP	If Z is the first gap in the sentence, NA; otherwise, if P_l is an NP node and P_r is a VP node, T; else F.	F
P_Is_VP	If Z is the first gap in the sentence, NA; otherwise, if P is a VP node, T; else F.	F
IP-VP	If in the path from W_r to C, there is a VP node such that its parent node is an IP node, T; else F.	T
Has_Ancestor_NP	If V has an NP node as ancestor, T; else F.	T
Has_Ancestor_VP	If V has a VP node as ancestor, T; else F.	F
Has_Ancestor_CP	If V has a CP node as ancestor, T; else F.	T
Left_Comma	If Z is the first gap, NA; otherwise if W_l is a comma, T; else F.	T
Subject_Role	If the grammatical role of Z is subject, S; else X.	X
Clause	If V is in a matrix clause, an independent clause, a subordinate clause, or none of the above, the value is M, I, S, X, respectively.	I
Is_In_Headline	If Z is in the headline of the text, T; else F.	F

Table 2: Features for anaphoric zero pronoun identification. The feature values of ϕ_2 are shown in the last column.

erate training examples from the training data set. All ZP candidates in the training data set generate training examples. Whether a training example is positive or negative depends on whether the ZP candidate is an AZP.

After generating all training examples, we train an AZP identification classifier using the J48 decision tree learning algorithm in Weka[5]. During testing, each ZP candidate is presented to the learned classifier to determine whether it is an AZP. We conduct experiments to measure the performance of the model learned. The results of 5-fold cross validation on the training data set are shown in Table 3.

Model	R	P	F
Heuristic	99.7	15.0	26.1
AZP Ident	19.8	51.1	28.6
AZP Ident ($r = 8$)	59.8	44.3	50.9

Table 3: Accuracies of AZP identification on the training data set under 5-fold cross validation.

We use heuristic rules as a baseline for comparison. The rules used by the heuristic model are as follows. For a node T in the parse tree, if

1. T is a VP node; and
2. T's parent node is not a VP node; and
3. T has no left sibling, or its left sibling is not an NP node,

then the gap that is immediately to the left of the word sequence spanned by T is an AZP. This simple AZP identification heuristic achieves an F-measure of 26.1%.

Imbalanced Training Data

From Table 3, one can see that the F-measure of the machine-learned AZP identification model is 28.6%, which is only slightly higher than baseline heuristic model. It has a relatively high precision, but much lower recall. The problem lies in the highly imbalanced number of positive and negative training examples. Among all the 155 texts in the training set, there are 343 positive and 10,098 negative training examples. The ratio r of the number

[5] `http://www.cs.waikato.ac.nz/ml/weka/`

of negative training examples to the number of positive training examples is 29.4. A classifier trained on such highly imbalanced training examples tends to predict more testing examples as negative examples. This explains why the precision is high, but the recall is low.

To overcome this problem, we vary r by varying the weight of the positive training examples, which is equivalent to sampling more positive training examples. The values of r that we have tried are $1, 2, 3, \ldots, 29$. The larger the value of r, the higher the precision, and the lower the recall. By tuning r, we get a balance between precision and recall, and hence an optimal F-measure. Figure 2 shows the effect of tuning r on AZP identification. When $r = 8$, the optimal F-measure is 50.9%, which is much higher than the F-measure without tuning r.

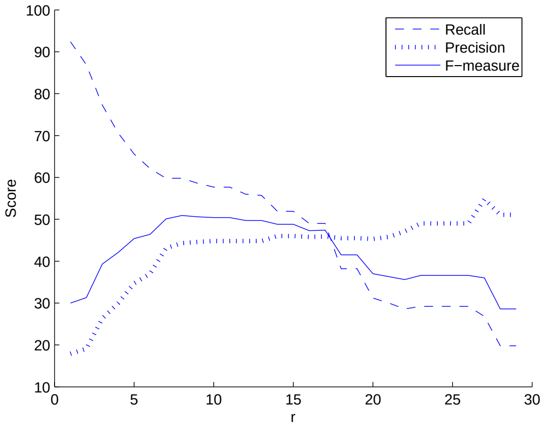


Figure 2: Effect of tuning r on AZP identification

Ng and Cardie (2002a) reported that the accuracies of their noun phrase anaphoricity determination classifier were 86.1% and 84.0% for the MUC6 and MUC7 data sets, respectively. Noun phrases provide much fruitful information for anaphoricity identification. However, useful information such as gender, number, lexical string, etc, is not available in the case of zero pronouns. This makes AZP identification a much more difficult task, and hence it has a relatively low accuracy.

5 Anaphoric Zero Pronoun Resolution

In anaphoric zero pronoun resolution, we also use machine learning techniques to build a classifier. The features are described in Table 4.

In the feature description, Z is the anaphoric zero pronoun that is under consideration, and A is the potential NP antecedent for Z. V is the same as in AZP identification. The feature values of the pair NP$_1$ and ϕ_2 (the gap immediately to the left of 占) in Figure 1 are shown in Table 4.

To train the AZP resolution classifier, we generate training examples in the following way. An AZP Z and its immediately preceding coreferential NP antecedent A in the gold standard coreference chain form a positive training example. Between A and Z, there are other NP candidates. Each one of these NP candidates, together with Z, form a negative training example. This is similar to the approach adopted in Soon et al. (2001). We also train the AZP resolution classifier using the J48 decision tree learning algorithm.

After building both AZP identification and resolution classifiers, we perform AZP identification and resolution in a sequential manner. For a ZP candidate Z, the AZP identification classifier determines whether Z is an AZP. If it is an AZP, all NP candidates that are to the left of Z in textual order are considered as potential antecedents. These potential antecedents are tested from right to left. We start from the NP candidate A_1 that is immediately to the left of Z. A_1 and Z form a pair. If the pair is classified as positive by the resolution classifier, A_1 is the antecedent for Z. If it is classified as negative, we proceed to the NP candidate A_2 that is immediately to the left of A_1, and test again. The process continues until we find an antecedent for Z, or there is no more NP candidate to test.

This right-to-left search attempts to find the closest correct antecedent for an AZP. We do not choose the best-first search strategy proposed by Ng and Cardie (2002b). This is because we generate training examples and build the resolution classifier by pairing each zero pronoun with its closest preceding antecedent. In addition, a zero pronoun is typically not too far away from its antecedent. In our data set, 92.6% of the AZPs have antecedents that are at most 2 sentences apart. Our experiment shows that this closest-first strategy performs better than the best-first strategy for Chinese AZP resolution.

Table 5 shows the experimental results of 5-fold cross validation on the training data set. For com-

Feature	Description	NP$_1$-ϕ_2
Features between Z and A		
Dist_Sentence	If Z and A are in the same sentence, 0; if they are one sentence apart, 1; and so on.	0
Dist_Segment	If Z and A are in the same segment (where a segment is a sequence of words separated by punctuation marks including “，”, “；”, “。”, “！”, and “？”), 0; if they are one segment apart, 1; and so on.	1
Sibling_NP_VP	If Z and A are in different sentences, F; Otherwise, if both A and Z are child nodes of the root node, and they are siblings (or at most separated by one comma), T; else F.	F
Closest_NP	If A is the closest preceding NP candidate to Z, T; else F.	T
Features on A		
A_Has_Anc_NP	If A has an ancestor NP node, T; else F.	F
A_Has_Anc_NP_In_IP	If A has an ancestor NP node which is a descendant of A's lowest ancestor IP node, T; else F.	F
A_Has_Anc_VP	If A has an ancestor VP node, T; else F.	F
A_Has_Anc_VP_In_IP	If A has an ancestor VP node which is a descendant of A's lowest ancestor IP node, T; else F.	F
A_Has_Anc_CP	If A has an ancestor CP node, T; else F.	F
A_Grammatical_Role	If the grammatical role of A is subject, object, or others, the value is S, O, or X, respectively.	S
A_Clause	If A is in a matrix clause, an independent clause, a subordinate clause, or none of the above, the value is M, I, S, X, respectively.	M
A_Is_ADV	If A is an adverbial NP, T; else F.	F
A_Is_TMP	If A is a temporal NP, T; else F.	F
A_Is_Pronoun	If A is a pronoun, T; else F.	F
A_Is_NE	If A is a named entity, T; else F.	F
A_In_Headline	If A is in the headline of the text, T; else F.	F
Features on Z		
Z_Has_Anc_NP	If V has an ancestor NP node, T; else F.	T
Z_Has_Anc_NP_In_IP	If V has an ancestor NP node which is a descendant of V's lowest ancestor IP node, T; else F.	F
Z_Has_Anc_VP	If V has an ancestor VP node, T; else F.	F
Z_Has_Anc_VP_In_IP	If V has an ancestor VP node which is a descendant of V's lowest ancestor IP node, T; else F.	F
Z_Has_Anc_CP	If V has an ancestor CP node, T; else F.	T
Z_Grammatical_Role	If the grammatical role of Z is subject, S; else X.	X
Z_Clause	If V is in a matrix clause, an independent clause, a subordinate clause, or none of the above, the value is M, I, S, X, respectively.	I
Z_Is_First_ZP	If Z is the first ZP candidate in the sentence, T; else F.	F
Z_Is_Last_ZP	If Z is the last ZP candidate in the sentence, T; else F.	F
Z_In_Headline	If Z is in the headline of the text, T; else F.	F

Table 4: Features for anaphoric zero pronoun resolution. The feature values of the pair NP$_1$ and ϕ_2 are shown in the last column.

parison, we show three baseline systems. In all three baseline systems, we do not perform AZP identification, but directly apply the AZP resolution classifier. In the first baseline, we apply the AZP resolution classifier on all ZP candidates. In the second baseline, we apply the classifier only on ZPs annotated in the gold standard, instead of all ZP candidates. In the third baseline, we further restrict it to resolve only AZPs. The F-measures of the three baselines are 2.5%, 27.6%, and 40.6% respectively.

Model	R	P	F
All ZP Candidates	40.5	1.3	2.5
Gold ZP	40.5	20.9	27.6
Gold AZP	40.5	40.6	40.6
AZP Ident (r=8 t=0.5)	23.6	17.5	20.1
AZP Ident (r=11 t=0.6)	22.4	20.3	21.3

Table 5: Accuracies of AZP resolution on the training data set under 5-fold cross validation.

Tuning of Parameters

Ng (2004) showed that an NP anaphoricity identification classifier with a cut-off threshold $t = 0.5$ pruned away many correct anaphoric NPs and harmed the overall recall. By varying t, the overall resolution F-measure was improved. We adopt the same tuning strategy and accept a ZP candidate ZP_i as an AZP and proceed to find its antecedent only if $P(ZP_i) \geq t$. The possible values for t that we have tried are $0, 0.05, 0.1, \ldots, 0.95$.

In Section 4, we show that $r = 8$ yields the best AZP identification F-measure. When we fix $r = 8$ and vary t, the overall F-measure for AZP resolution is the best at $t = 0.65$, as shown in Figure 3. We then try tuning r and t at the same time. An overall optimal F-measure of 21.3% is obtained when $r = 11$ and $t = 0.6$. We compare this tuned F-measure with the F-measure of 20.1% at $r = 8$ and $t = 0.5$, obtained without tuning t. Although the improvement is modest, it is statistically significant ($p < 0.05$).

6 Experimental Results

In the previous section, we show that when $r = 11$ and $t = 0.6$, our sequential AZP identification and resolution achieves the best F-measure under 5-fold cross validation on the 155 training texts. In order to utilize all available training data, we generate training examples for the AZP identification classifier with $r = 11$, and generate training examples for the AZP resolution classifier, on all 155 training texts. Both classifiers are trained again with the newly generated training examples. We then apply both classifiers with anaphoricity identification cut-off threshold $t = 0.6$ to the blind test data. The results are shown in Table 6.

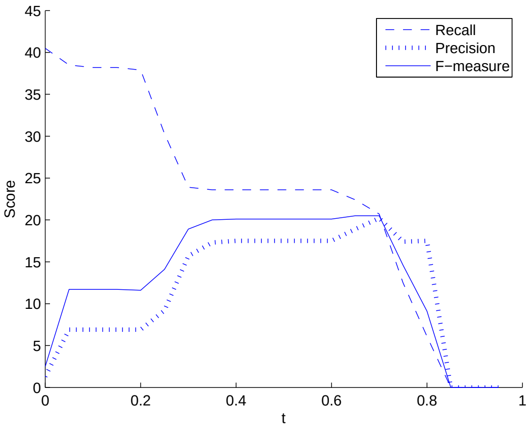


Figure 3: Effect of tuning t on AZP resolution

R	P	F
27.5	24.4	25.9

Table 6: Accuracies of AZP resolution on blind test data.

By utilizing all available information on the gold standard parse trees, Converse (2006) finds an antecedent for each AZP given that all AZPs are correctly input to her system. The accuracy of her rule-based approach is 43.0%. For comparison, we determine the antecedents for AZPs in the gold standard annotation, under 5-fold cross validation on all 205 texts in the corpus. The recall, precision, and F-measure are 42.3%, 42.7%, and 42.5%, respectively. This shows that our proposed machine learning approach for Chinese zero pronoun resolution is comparable to her state-of-the-art rule-based approach.

7 Related Work

Converse (2006) assumed that the gold standard Chinese anaphoric zero pronouns and the gold standard parse trees of the texts in Penn Chinese Tree-

Bank (CTB) were given as input to her system, which performed resolution of the anaphoric zero pronouns using the Hobbs algorithm (Hobbs, 1978). Her system did not identify the anaphoric zero pronouns automatically.

Yeh and Chen (2004) proposed an approach for Chinese zero pronoun resolution based on the Centering Theory (Grosz et al., 1995). Their system used a set of hand-engineered rules to perform zero pronoun identification, and resolved zero pronouns with a set of hand-engineered resolution rules.

In Iida et al. (2006), they proposed a machine learning approach to resolve zero pronouns in Japanese using syntactic patterns. Their system also did not perform zero pronoun identification, and assumed that correctly identified zero pronouns were given as input to their system.

The probabilistic model of Seki et al. (2002) both identified and resolved Japanese zero pronouns, with the help of a verb dictionary. Their model needed large-scale corpora to estimate the probabilities and to prevent data sparseness.

Ferrández and Peral (2000) proposed a hand-engineered rule-based approach to identify and resolve zero pronouns that are in the subject grammatical position in Spanish.

8 Conclusion

In this paper, we present a machine learning approach to the identification and resolution of Chinese anaphoric zero pronouns. We perform both identification and resolution automatically, with two sets of easily computable features. Experimental results show that our proposed learning approach achieves anaphoric zero pronoun resolution accuracy comparable to a previous state-of-the-art, heuristic rule-based approach. To our knowledge, our work is the first to perform both identification and resolution of Chinese anaphoric zero pronouns using a machine learning approach.

Obviously, there is much room for improvement. In future, we plan to apply our model directly on machine-generated parse trees. We also plan to classify non-coreferential zero pronouns into the six categories.

Acknowledgements

We thank Susan Converse and Martha Palmer for sharing their Chinese third-person pronoun and zero pronoun coreference corpus.

References

Susan Converse. 2006. *Pronominal Anaphora Resolution in Chinese.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Antonio Ferrández and Jesús Peral. 2000. A computational approach to zero-pronouns in Spanish. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, pages 166–172.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics Annual Meeting 2004 (HLT-NAACL2004)*, pages 1–8.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Jerry R. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311–338.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL2006)*, pages 625–632.

Young-Joo Kim. 2000. Subject/object drop in the acquisition of Korean: A cross-linguistic comparison. *Journal of East Asian Linguistics*, 9(4):325–351.

Charles N. Li and Sandra A. Thompson. 1979. Third-person pronouns and zero-anaphora in Chinese discourse. *Syntax and Semantics*, 12:311–335.

Wendan Li. 2004. Topic chains in Chinese discourse. *Discourse Processes*, 37(1):25–45.

Andrew McCallum and Ben Wellner. 2005. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 905–912.

Vincent Ng and Claire Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING2002)*, pages 1–7.

Vincent Ng and Claire Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL2002)*, pages 104–111.

Vincent Ng. 2004. Learning noun phrase anaphoricity to improve coreference resolution: Issues in representation and optimization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL2004)*, pages 152–159.

Kazuhiro Seki, Atsushi Fujii, and Tetsuya Ishikawa. 2002. A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING2002)*, pages 911–917.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL2003)*, pages 176–183.

Ching-Long Yeh and Yi-Chun Chen. 2004. Zero anaphora resolution in Chinese with shallow parsing. *Journal of Chinese Language and Computing*.

Yaqian Zhou, Changning Huang, Jianfeng Gao, and Lide Wu. 2005. Transformation based Chinese entity detection and tracking. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP 2005)*, pages 232–237.

Parsimonious Data-Oriented Parsing

Willem Zuidema
Institute for Logic, Language and
Computation, University of Amsterdam
Plantage Muidergracht 24
1018 TV, Amsterdam, the Netherlands
jzuidema@science.uva.nl

Abstract

This paper explores a parsimonious approach to Data-Oriented Parsing. While allowing, in principle, all possible subtrees of trees in the treebank to be productive elements, our approach aims at finding a manageable subset of these trees that can accurately describe empirical distributions over phrase-structure trees. The proposed algorithm leads to computationally much more tracktable parsers, as well as linguistically more informative grammars. The parser is evaluated on the OVIS and WSJ corpora, and shows improvements on efficiency, parse accuracy and testset likelihood.

1 Data-Oriented Parsing

Data-Oriented Parsing (DOP) is a framework for statistical parsing and language modeling originally proposed by Scha (1990). Some of its innovations, although radical at the time, are now widely accepted: the use of fragments from the trees in an annotated corpus as the symbolic grammar (now known as "treebank grammars", Charniak, 1996) and inclusion of all statistical dependencies between nodes in the trees for disambiguation (the "all-subtrees approach", Collins & Duffy, 2002).

The best known instantiations of the DOP-framework are due to Bod (1998; 2001; 2003), using the Probabilistic Tree Substitution Grammar (PTSG) formalism. Bod has advocated a *maximalist* approach to DOP, inducing grammars that contain all subtrees of all parse trees in the treebank, and using them to parse unknown sentences where all of these subtrees can potentially contribute to the most probable parse. Although Bod's empirical results have been excellent, his maximalism poses important computational challenges that, although not necessarily unsolvable, threaten both the scalability to larger treebanks and the cognitive plausibility of the models.

In this paper I explore a different approach to DOP, that I will call "Parsimonious Data-Oriented Parsing" (P-DOP). This approach remains true to Scha's original program, by allowing, in principle, all possible subtrees of trees in the treebank to be the productive elements. But unlike Bod's approach, P-DOP aims at finding a succinct subset of such elementary trees, chosen such that it can still accurately describe observed distributions over phrase-structure trees. I will demonstrate that P-DOP leads to computationally more tracktable parsers, as well as linguistically more informative grammars. Moreover, as P-DOP is formulated as an enrichment of the treebank Probabilistic Context-free Grammar (PCFG), it allows for much easier comparison to alternative approaches to statistical parsing (Collins, 1997; Charniak, 1997; Johnson, 1998; Klein and Manning, 2003; Petrov et al., 2006).

2 Independence Assumptions in PCFGs

Parsing with treebank PCFGs, in its simplest form, involves the following steps: (1) a treebank is created by extracting phrase-structure trees from an annotated corpus, and split in a train- and a testset; (2) a PCFG is read off from all productions in the trainset trees, with weights proportional to their fre-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 551–560, Prague, June 2007. ©2007 Association for Computational Linguistics

quency in the treebank (the "relative frequency estimate"); (3) a standard PCFG parser is used to find for each yield of the test-set trees the most probable parse; (4) these parses are compared to the test-set trees to count matching brackets, labels and trees.

PCFGs incorporate a strong statistical independence assumption: that the expansion of a nonterminal node is only dependent on the node's label. All state-of-the-art wide-coverage parsers relax this assumption in some way, for instance by (i) changing the parser in step (3), such that the application of rules is *conditioned* on other steps in the derivation process (Collins, 1997; Charniak, 1997), or by (ii) *enriching* the nonterminal labels in step (1) with context-information (Johnson, 1998; Klein and Manning, 2003), along with suitable backtransforms in step (4). These two approaches often turn out to be equivalent, although for some conditionings it is not trivial to work out the equivalent enrichment and vice versa, especially when combined with smoothing. Interesting recent work has focused on the automatic induction of enrichments (Matzuzaki et al., 2005; Prescher, 2005), leading to extremely accurate parsers (Petrov et al., 2006).

DOP relaxes the independence assumption by changing the class of probabilistic grammars induced in step (2). In DOP1 (Bod, 1998), a PTSG is induced, which consists, subject to some heuristic constraints, of all subtrees[1] of the treebank trees with a weight proportional to their frequency. PTSGs allow multiple derivations to yield the same parse; in DOP1 the sum of their probabilities gives the probability of the parse. The relation between DOP and enrichment/conditioning models was clarified by Joshua Goodman, who devised an efficient PCFG transform of the DOP1 model (Goodman, 1996). The size of the PCFG resulting from this transform is linear in the number of nonterminals tokens in the corpus. Goodman's transform, in combination with a range of heuristics, allowed Bod (2003) to run the DOP model on the Penn Treebank WSJ benchmark and obtain some of the best results obtained with a generative model.

The computational challenges for DOP are far from solved, however. The difference with style (ii) enrichment is that we derive many more rules from every original tree than the number of CFG-productions it contains. This is one reason why the relative frequency estimator for DOP is inconsistent (Johnson, 2002). But worse, perhaps, the size of the grammar remains gigantic[2], making it difficult for many in the field to replicate Bod's results.

In this paper, we develop a parsimonious approach to DOP, that avoids many of the computational problems of the maximalist approach but tries to maintain its excellent empirical performance. Our approach starts, both conceptually and technically, with an analysis of where the PCFG independence assumption breaks down when modeling empirical distributions. In section 2 we derive equations for the expected frequency of arbitrary subtrees under a distribution defined by a given PCFG, and use them to measure how much observed subtree-frequencies deviate from expectation. In section 4 we generalize this analysis to PTSGs. In section 5 we discuss an algorithm for estimating PTSGs from a treebank, that is based on minimizing the differences between expected and observed subtree-frequencies. We then proceed with discussing PTSGs induced from various treebanks, and in section 6 the use of these PTSGs for parsing.

3 Deviations from a PCFG distribution

PCFGs can be viewed as PTSGs where the elementary trees are restricted to depth 1; we therefore start by repeating the definition of PTSGs (Bod, 1998), and use notation appropriate for PTSGs throughout. An PTSG is a 5-tuple $\langle V_n, V_t, S, T, w\rangle$, where V_n is the set of non-terminal symbols; V_t is the set of terminal symbols; $S \in V_n$ is the start symbol; T is a set of elementary trees, such that for every $\tau \in T$ the unique root node $r(\tau) \in V_n$, the (possibly empty) set of internal nodes $i(\tau) \subset V_n$ and the set of leaf nodes $l(\tau) \subset V_n \cup V_t$; finally, $w : T \rightarrow [0, 1]$ is a probability (weight) distribution over the elementary trees, such that for any $\tau \in T$, $\sum_{\tau' \in R(\tau)} w(\tau') = 1$, where $R(\tau)$ is the set of elementary trees with the same root label as τ. It will prove useful to also define the set of all possible trees θ over the defined

[1] A subtree t' of a parse tree t is a tree such that every node i' in t' equals a node i in t, and i' either has no daughters or the same daughter nodes as i.

[2] Sections 2-21 of WSJ contain 1676821 productions. Of these,10^6 are lexical productions, and 36151 top-productions, leaving approx. 640000 internal productions which yield about 2.5×10^6 rules in Goodman's transform.

alphabets (with the same conditions on root, internal and leaf nodes as for T), and the set of all possible *complete* parse trees Θ (with $r(t) = S$ and all leaf nodes $l(t) \subset V_t$). Obviously, $T \subset \theta$ and $\Theta \subset \theta$.

The substitution operation $\circ$ is defined if the *leftmost nonterminal leaf* in τ_1 is identical to the root of τ_2. Performing substitution $\tau_1 \circ \tau_2$ yields t_3, if t_3 is identical to τ_1 with the leftmost nonterminal leaf replaced by τ_2. A derivation is a sequence of elementary trees, where the first tree $\tau \in T$ has root-label S and every next tree combines through substitution with the result of the substitutions before it. In this paper, we are only concerned with grammars that define proper probability distributions over trees, such that the probability of all derivations sum up to 1 and no probability mass gets lost in derivations that never reach a terminal yield. That is, we require (if $t(d)$ is the tree derived by derivation d):

$$\sum_{d:t(d)\in\Theta} P(d) = 1. \tag{1}$$

For simplicity, but without loss of generality, we assume there are no recursions on the start symbol.

In this section, we restrict ourselves to PCFG distributions, and thus to a T with only depth 1 trees. The probability of a PCFG rule (conditioned on its left-hand side) in the conventional notation, $P(A \mapsto \alpha\beta\ldots\gamma|A)$, now corresponds to the probability of a depth 1 tree (conditioned on its root nonterminal):

$$P\left(\begin{array}{c} \text{A} \\ \alpha \quad \beta \quad \ldots \quad \gamma \end{array} |A\right)$$

Of course, the probability of a (complete) derivation is simply the product of the (conditional) probabilities of the rules in the derivation. It is useful to consider, for a given grammar G generating a corpus of N trees, the expected frequency of visiting nonterminal state X:

$$\mathcal{E}F(X) = \begin{cases} N & \text{if } X = S \\ \sum_\tau \mathcal{E}F(\tau)C(X, l(\tau)) & \text{otherwise} \end{cases} \tag{2}$$

where $C(X, l(\tau))$ gives the number of occurrences of nonterminal X among the leaves of elementary tree τ. Furthermore, the expected *usage frequency* of τ is given by

$$\begin{aligned} \mathcal{E}F(\tau) &= \mathcal{E}F(r(\tau))P(\tau|r(\tau)) \\ &= \mathcal{E}F(r(\tau))w(\tau) \end{aligned} \tag{3}$$

Substituting eq (3) into (2) yields a system of $|V_n|$ linear equations, that can be straightforwardly solved using standard methods.

We are interested in the empirical deviations from the distribution defined by a given grammar (for instance, the treebank PCFG), such that we can adjust the grammar to better model the training data (whilst avoiding overfitting). In line with the general DOP approach, we would like to measure this deviation for every possible subtree. Of course, the conditional probability of an arbitrary subtree is simply the product of the rule probabilities. The expected frequency of a subtree is the expected frequency of its root state, times the conditional probability:

$$\mathcal{E}F(t) = \mathcal{E}F(r(t))P(t|r(t)) \tag{4}$$

Using these equations, we can measure for each observed subtree in the corpus, the difference between observed frequency and expected frequency. This will give high values for overrepresented and frequent constructions in the corpus, such as subtrees corresponding to *revenues rose CD % to \$ CD million from \$ CD million last year*, *details weren't disclosed*, *NP-SUBJ declined to comment* and contracted and negated auxiliaries such as *won't*, *can't* and *don't*. The top-10 overrepresented subtrees in the WSJ20-corpus are given in figure 1.

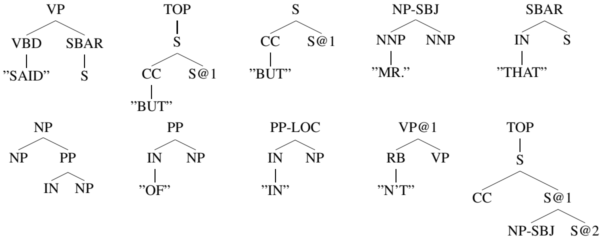


Figure 1: Top-10 overrepresented subtrees (excluding subtrees with punctuation) in sentences of length ≤ 20, including punctuation, in sections 2-21 of the WSJ-corpus (transformed to Chomsky Normal Form, whereby newly created nonterminals are marked with an @). Measured are the deviations from the expected frequencies according to the treebank PCFG (of this selection), as in equation (4) but with $\mathcal{E}F(r(t))$ replaced by the empirical frequency $o(r(t))$. Observed frequencies are (deviations between brackets): 461 (+408.2), 554 (+363.8), 556 (+361.7), 479 (+348.2), 332 (+314.3), 415 (+313.3), 460 (+305.1), 389 (+283.0), 426 (+277.2), 295 (+266.1).

Of course, there are also many subtrees that occur much less frequently than the grammar predicts, such as for instance subtrees corresponding to infrequent or non-occurring variations of the frequent ones, e.g. *revenues rose CD from $ CD million from $ CD million*. Underrepresented subtrees found in the WSJ20 corpus, include (VP (VBZ ”IS”) NP)), which occurs only once, even though it is predicted 152.7 times more often (in all other VP's with “IS”, the NP is labeled NP-PRD); and (PP (IN ”IN”) NP)), which occurs 38 times but is expected 121.0 times more often (IN NP-constructions are usually labeled PP-LOC).

Given such statistics, how do we improve the grammar such that it better models the data? PCFG enrichment models (Klein and Manning, 2003; Schmid, 2006) split (and merge) nonterminals; in automatic enrichment methods (Prescher, 2005; Petrov et al., 2006) these transformations are performed so as to maximize data likelihood (under some constraints). The treebank PCFG-distribution thereby changes, such that the deviations from figure 1 mostly disappear. For instance, the overrepresentation of “but” as the sentence-initial CC in the second and third subtree of that figure, is dealt with in (Schmid, 2006) by splitting the CC-category into CC/BUT and CC/AND. However, also when a range of such transformations is applied, some subtrees are still greatly overrepresented. Figure 2 gives the top-10 overrepresented subtrees of the same treebank, enriched with Schmid's enrichment program `tmod`.

In DOP, larger subtrees can be explicitly represented as units. This is the approach we take in this paper, which involves switching from PCFGs to PTSGs. However, we cannot simply add overrepresented trees to the treebank PCFG; as is clear from figure 2, many of the overrepresented subtrees are in fact spurious variations of the same constructions (e.g. “$ CD million”, “a JJ NN”). To reach our goal of finding the minimal set of subtrees that accurately models the empirical distribution over trees, we will thus need to consider a series of PTSGs, find the subtrees that are still overrepresented and adapt the grammar accordingly.

4 Deviations from an PTSG distribution

4.1 Expected Frequencies: An Example

Once we allow T to contain elementary trees of depth larger than 1, the equations above become more difficult. The reason is that now multiple derivations may give rise to the same parse tree, and, as a corrolary, a specific subtree can emerge in many different ways. Consider an PTSG that consists of all subtrees of the trees t_1, t_2 and t_3 in figure 3, and the expected frequency of the subtree t^*.

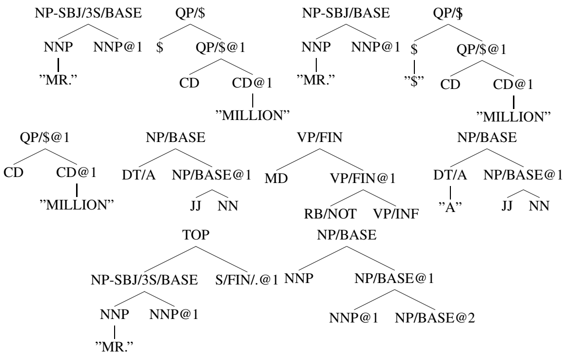


Figure 2: Top-10 overrepresented subtrees (excluding subtrees with punctuation) in the WSJ20 corpus, enriched with the `tmod` program (Schmid, 2006). Empirical frequencies are as follows (deviations between brackets): 262 (+207.6), 235 (+158.4) 207 (+156.4), 228 (+153.5), 237 (+141.0), 190 (+134.2), 153 (+126.5), 166 (+117.8), 139 (+110.0), 111 (+103.8).

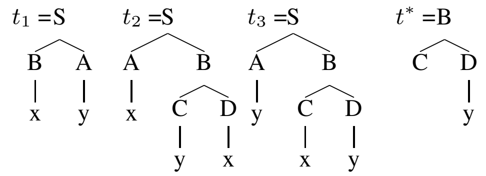


Figure 3: Three example treebank trees and the focal subtree

It is clear that t^* might arise in many different ways. For instance, it emerges in the derivation with elementary trees $\tau_1 \circ \tau_4 \circ \tau_5$ from figure 4, but also in derivations $\tau_2 \circ \tau_4$ and $\tau_3 \circ \tau_5$. Note that in none of these derivations elementary tree t^* itself was used.

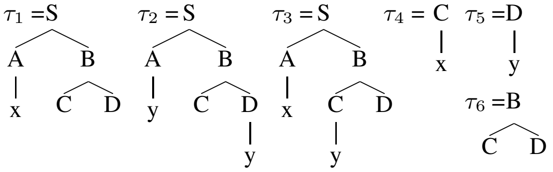


Figure 4: Some elementary trees extracted from the trees in fig 3

4.2 Expected Frequency: Usage & Occurrence

Hence, when using PTSGs, we need to distinguish between the *expected usage frequency* of an elementary tree (written as $\mathcal{E}u(\tau)$), and the *expected occurrence frequency* ($\mathcal{E}o(t)$) of the corresponding subtree. Moreover, not all nonterminal nodes in a derived tree are necessarily “visited” substitution sites. The expected frequency of visiting a nonterminal state X as substitution site depends on the usage frequencies:

$$\mathcal{E}F(X) = \begin{cases} N & \text{if } X = S \\ \sum_{\tau} \mathcal{E}u(\tau) C(X, l(\tau)) & \text{otherwise} \end{cases} \tag{5}$$

Relating usage frequencies to weights is still simple (compare equation 3):

$$\mathcal{E}u(\tau) = \mathcal{E}F(r(\tau)) w(\tau) \tag{6}$$

And hence: $w(\tau) = \mathcal{E}u(\tau) / \sum_{\tau' : r(\tau) = r(\tau')} \mathcal{E}u(\tau')$.

The expected frequency of a complete tree is not simply a product anymore, but the sum of the different derivation probabilities (where der(t) gives the set of derivations of t):

$$\mathcal{E}o(t) = \sum_{d \in \text{der}(t)} \prod_{\tau \in d} w(\tau) \qquad \text{if } t \in \Theta \tag{7}$$

4.3 Expected Frequency of Arbitrary Subtrees

Most complex is the expected occurrence frequency of an arbitrary subtree t. From the example above it is clear that it is not necessary that the root of t is a substitution site. Analogous to equation (4), we need the expected frequency of arriving at some state σ in the derivation process that is still consistent with expanding to something that contains t, and multiply it with the probability that this expansion indeed happens:

$$\mathcal{E}o(t) = \sum_{\sigma} \mathcal{E}F(\sigma) P(t|\sigma) \tag{8}$$

To be able to define the states σ, we redefine the set of derivations $der(t)$ of a subtree t, such that the derivations $der(t^*)$ of our example tree from figure 3 are the following: $d_1 = B \circ \tau_6 \circ \tau_5$, $d_2 = \tau_6 \circ \tau_5$, $d_3 = B \circ t^*$ and $d_4 = t^*$. Only if a derivation starts with a single nonterminal is the root node considered a substitution site. The states σ correspond to the first elements of each of these derivations, i.e. $\langle B, \tau_6, B, t^* \rangle$.

As was clear from the example in section 4.1, we need to consider all supertrees of the trees in the derivation of t for calculating the expected frequency of a state and the probability of expanding from that state to form t. It is useful to distinguish, as do Bod & Kaplan (Bod, 1998, ch. 10) two types of supertree-subtree relations, depending on whether nodes must be removed from the root downward, or from the leaves (“frontier”) upward. “Root-subtrees” of t are those subtrees headed by any of t’s internal nodes and everything below. “Frontier-subtrees” are those subtrees headed by t’s root-node, pruned at any number (≥ 0) of internal nodes. Using $\circ$ to indicate left-most substitution, we can write:

- t_1 is a *root-subtree* of t_1, and t_1 is a *root-subtree* of t_2, if $\exists t_3$, such that $t_3 \circ t_1 = t_2$;
- t_1 is a *frontier-subtree* of t_1, and t_1 is a *frontier-subtree* of t_2, if $\exists t_3 \ldots t_n$, such that $t_1 \circ t_3 \ldots \circ t_n = t_2$.
- t' is the x-frontier-subtree of t, $t' = fs_x(t)$, if x is a set of nodes in t, such that if t is pruned at each $i \in x$ it equals t'.

We use the notation $st(t)$ for the set of subtrees of t, $rs(t)$ for the set of root-subtrees of t and $fs(t)$ for the set of frontier-subtrees of t. Thus defined, the set of all subtrees of t is the set of all frontier-subtrees of all root-subtrees of t: $st(t) = \{t' | \exists t'' (t'' \in rs(t) \wedge t' \in fs(t''))$. We further define the sets of root-supertrees, frontier-supertrees, and supertrees as follows: (i) $\widehat{fs_x}(t) = \{t' | t = fs_x(t')\}$, (ii) $\widehat{fs}(t) = \{t' | t \in fs(t')\}$ (iii) $\widehat{st}(t) = \{t' | t \in st(t')\}$.

If there are only terminals in the yield of t, the expected frequency of a state σ is now simply the sum of the expected usage frequencies of those elementary trees that have σ at their *frontier* (i.e. that σ is a root-subtree of):

$$\mathcal{E}F(\sigma) = \sum_{\tau' : \sigma \in rs(\tau')} \mathcal{E}u(\tau') \qquad \text{if } l(t) \subset V_t \tag{9}$$

If there are nonterminals in the yield of t, as in the example, we need to also consider elementary trees that have these nonterminals already expanded. To see why, consider again the example of section 4.1 and check that also elementary tree τ_3 contributes to the expected frequency of t^*. If we take this into account, and write $nt(t)$ for the nonterminal nodes in the yield of t, the final expression for the expected frequency of state σ becomes:

$$\mathcal{E}F(\sigma) = \sum_{\tau \in \widehat{fs}(\sigma)} \sum_{\tau' \in \widehat{rs_{nt(t)}}(\tau)} \mathcal{E}u(\tau') \tag{10}$$

Finally, the probability of expanding a state σ such that t emerges is again simplest if t has no nonterminals as leaves. Remember that a state σ was the first element of a derivation of t; the probability of expanding to t is simply the product of the weights of the remaining elementary trees in the derivation (if states are unique for a derivation):

$$P(t|\sigma) = \prod_{\tau \in \text{rest}(d)} w(\tau) \qquad \text{if } l(t) \subset V_t \quad (11)$$

If there are nonterminals among the leaves of t, however, we need again to sum over possible expansions at those nonterminal leaves:

$$P(t|\sigma) = \prod_{\tau \in \text{rest}(d)} \sum_{\tau' \in \widehat{fs}_{x(t)}(\tau)} w\left(\tau'\right) \qquad (12)$$

Substituting equations (9) and (12) into equation (8) gives a general expression for the expected occurrence frequency of an arbitary subtree t:

$$\mathcal{E}o(t) = \sum_{d \in der(t)} \left(\prod_{\substack{\tau \in \\ \text{rest}(d)}} \sum_{\tau' \in \widehat{fs}_{x(t)}(\tau)} w\left(\tau'\right) \right.$$
$$\left. \sum_{\substack{\tau \in \\ \widehat{rs}(\text{first}(d))}} \sum_{\tau' \in \widehat{fs}_{x(t)}(\tau)} \mathcal{E}u(\tau') \right). \qquad (13)$$

5 Minimizing deviations: estimation

The equations just derived can be used to learn an PTSG from a treebank, using an estimation procedure we call "push-n-pull" (pnp). This procedure was described in some detail elsewhere (Zuidema, 2006b); here I only sketch the basic idea. Given an initial setting of the parameters (all depth 1 elementary trees at their empirical frequency), the method calculates the expected frequency of all complete and incomplete trees. If a tree t's expected frequency $\mathcal{E}o(t)$ is higher than its observed frequency $o(t)$, the method subtracts the difference from the tree's score, and distributes ("pushes") it over the elementary trees involved in all its derivations ($der(t)$). If it is lower, it "pulls" the difference from all its derivations.

The "score" of an elementary tree τ is the algorithm's estimate of the usage frequency $u(\tau)$. The amounts of score that are pushed or pulled are capped by the requirement that $\forall \tau \; u(\tau) \geq 0$; moreover, the learning rate parameter γ determines the fraction of the expected-observed difference that is actually pushed or pulled. Finally, the method includes a bias (B) for moving probability mass to smaller elementary trees, to avoid overfitting (its effects become smaller as more data gets observed).

Because smaller elementary trees will be involved in other derivations as well, the push and pull operations will shift probabilities between different parse trees. Suppose a given complete tree is the only tree with nonzero frequency of all trees that can be built from the same components. This tree will continue to "pull" until it has in fact reached its appropriate frequency. Similarly, if a given tree does have zero observed frequency, it will continue to leak score to other derivations with the same components.

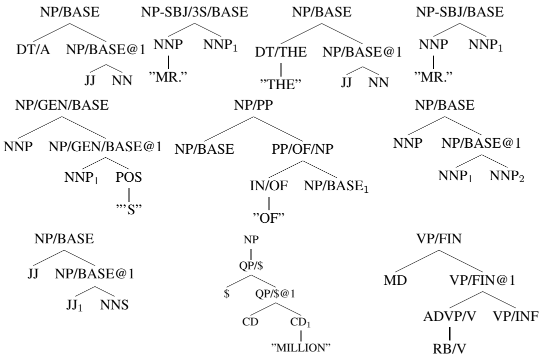


Figure 5: Top-10 elementary trees of depth>1, excluding those with punctuation, from running pnp on the enriched WSJ20.

The output of the push-n-pull algorithm is an PTSG, with the same set of elementary trees as the DOP models of Bod (1998; 2001). This set is very large. However, unlike those existing DOP models, the score distribution over these elementary trees is extremely skewed: relatively few trees receive high scores, and there is a very long tail of trees with low scores. In Zuidema (2006b) we give a qualitative analysis of the subtrees with the highest scores as induced from the ATIS treebank, which include many of its frequent constructions including *show me NP*, *I would like NP*, *flights from NP to NP*. The top-10 larger elementary trees that result from running pnp on a randomly selected trainset of about 8000 sentences of the Dutch OVIS treebank (Veldhuijzen van Zanten et al., 1999), can be glossed as: *Yes, from NP to NP, No thank you very much, I want to VP-INF,* *No thank you, I want PP to VP-INF, I want PP, I want Prep NP-LOC Prep NP-LOC, Yes please, At CD o'clock*. In figure 5 we give the top-10 elementary trees resulting from the WSJ20-corpus.

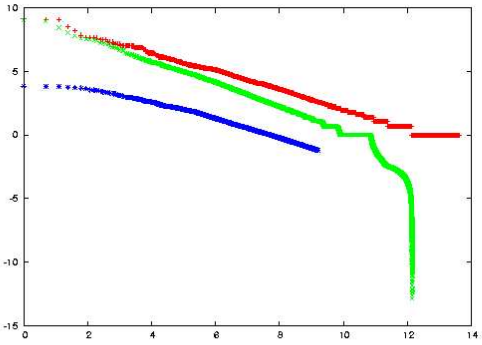


Figure 6: Log-log curves of (i) subtree frequencies against rank (for 10^6 subtrees from WSJ20), (ii) pnp-scores against rank, and (iii) the same for the top-10000 depth>1-subtrees.

Figure 6 shows some characteristics of this last grammar. Shown are log-log plots, such as commonly used to visualise the Zipf-distributions in natural language. The top curve plots log(frequency) against log(rank) for each subtree of the trees in the corpus, which shows the approximate Zipfian behavior. The second curve from above plots the log(score) against log(rank) for these same subtrees. As can be observed, the score-distribution follows the frequency distribution only for the most frequent subtrees (all of depth 1), but then deviates from it downwards. The bottom curve – an almost straight line in this log-log space – gives the log(score) vs log(rank) of subtrees with a depth>1.

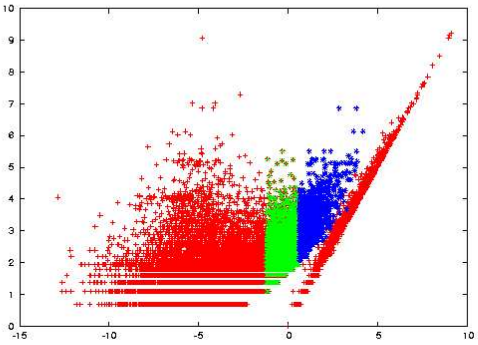


Figure 7: Subtree frequencies against pnp-scores, including subsets pnp1000 (dark/blue) and pnp10000 (light/green).

Figure 7 further illustrates the difference between the score- and the frequency-distributions, by plotting for each subtree, log-frequency (y-axis) against log-score (x-axis). The subtrees clearly fall into two categories: those where the scores correlate strongly with frequency (the depth 1 subtrees) and the larger subtrees that vary greatly in how strong scores correlate with frequency. Only larger subtrees that receive relatively high scores should be used in parsing.

Weights are proportional to subtree-frequencies in the DOP1 and related "maximalist" models. The differences between the frequency and score-distributions thus illustrate a very important difference between maximalist and parsimonious DOP. The characteristics of the score distribution allow P-DOP to throw away most of the subtrees without significantly affecting the distribution over complete parse trees that the grammar defines. This is the approach we take for evaluating parsing performance: we take as our baseline the treebank PCFG, and then add the n larger elementary trees with the highest scores from our induced PTSG.

6 Parsing Results

For our parsing results we use BitPar, a fast and freely available general PCFG parser (Schmid, 2004). In our first experiments we used the OVIS corpus, with semantic tags and punctuation removed, and all trees (train- and testset) binarized. As a baseline experiment, we read off the treebank PCFG as decribed in section 2. The recall, precision and complete match results are in table 1, labeled tb-pcfg. For comparison, we also show the results obtained with two versions of the DOP model, DOP1 (Bod, 1998) and DOP* (Zollmann and Sima'an, 2005) on the same treebank.

We ran the pnp program as described above on the trainset, with parameters $B = 1.0$, $\gamma = 0.1$ and $d = 4$. This run yielded a single PTSG that was used in 4 parsing runs. For these experiments, we added increasingly many of the depth>1 elementary trees from the PTSG, with minimum scores of 7.0, 1.0, 0.5, and 0.075. The added elementary trees were first converted to PCFG rules, by labeling all internal nodes with a unique address label and reading off the CFG-productions. Each rule received a score equal to the score of the elementary tree it derived from. A copy of each rule, with the label removed, was also added with a negative score, BitPar automatically sums (and substracts) and normalizes the frequency information provided with each rule. BitPar was then run on the testset sentences, with the option to output the n best parses with $n = 10$ by default. These parses were then read in in a post-processing program, which removes address labels, sums probabilities of equivalent parses and outputs the most probable parse for each sentence (this is the same approximation of MPP, albeit with smaller n, as used in most of Bod's DOP results). The results of these experiments are also in table 1, labeled pnp$N$, where N is the number of elementary trees added.

model	# rules	LR	LP	CM
tb-pcfg	3000	93.45	95.5	85.84
DOP1	1.4×10^6			(87.55)
DOP*	(< 50000)			(87.7)
pnp100	3000+100	93.63	95.65	86.55
pnp763	3000+763	93.5	95.52	86.75
pnp1517	3000+1517	93.78	95.83	87.36
pnp11411	3000+11411	**94.26**	**96.4**	**87.77**

Table 1: Results on the Dutch OVIS tree bank, with semantic tags and punctuation removed. Reported are evalb scores on a random testset of 1000 sentences (a second testset of 1000 sentences is kept for later evaluations). The trainset for both the treebank grammar and the pnp program consists of the remaining 8049 trees. Coverage in all cases in 989 sentences out of 1000. Results in brackets are from Zollman & Sima'an, 2005, using a different train-test set split.

As these experiments show, adding larger elementary trees from the induced PTSG, in order of their assigned scores, monotonously increases the parse accuracy of the treebank PCFG. Although the final grammar is at least 5 times larger than the original treebank PCFG, and the parser therefore slower, the grammar is orders of magnitude smaller than the corresponding maximalist DOP models and shows comparable parse accuracy.

For a second set of parsing experiments, we used the WSJ portion of the Penn Tree Bank (Marcus et al., 1993) and Helmut Schmid's enrichment program tmod (Schmid, 2006). Schmid's program enriches nonterminal labels in the treebank, using features inspired by (Klein and Manning, 2003). After enrichment, Schmid obtained excellent parsing scores with the treebank PCFG. In table 2, as model tb-pcfg, we give our baseline results. These are slightly lower than Schmid's, for two reasons: (i) our implementation ignores the upper/lower case distinction, and (ii) we do not use Schmid's word class automaton for unknown words (the only smoothing used is the built-in feature of the BitPar parser, which extracts an open-class-tag file from the lexicon file). Because our interest here is in the principles of enrichment we have not attempted to adapt these techniques for our implementation.

As before, we ran the pnp program on the trainset, the enriched sections 2-21 of the WSJ. For computational reasons, pnp is only run on trees with a yield of length (including punctuation) ≤ 20. This run, which took several days on a machine with 1.5Gb RAM, again produced a very large PTSG, from which we extracted the 1000 and 10000 depth>1 elementary trees with the highest scores for parsing experiments. Parsing and postprocessing is performed as before, with the MPP approximated from the best $n = 20$ parses. Results from these experiments are shown in table 2, as models pnp1000 and pnp10000. With a small number of added trees, we see a small drop in the parsing performance, which we interpret as evidence that our additions somewhat disturb the nicely tuned probability distribution of the treebank PCFG without providing many advantages, because the most frequent constructions have already been addressed in the manual PCFG enrichment. However, with 10000 added subtrees we see an increase in parse accuracy, providing evidence that pnp has learned potential enrichments that go beyond the manual enrichment.

model	LR	LP	F_1	CM
tb-pcfg	83.27	83.53	83.40	26.58
pnp1000	83.20	83.47	83.33	26.70
pnp10000	**83.56**	**83.99**	**83.77**	**26.93**

Table 2: Results on the WSJ section of the Penn Tree Bank, where nonterminals are enriched with features using Helmut Schmid's tmod program (Schmid, 2006). Reported are evalb scores (ignoring punctuation) on 1699 sentences ≤ 100, including punctuation, from section 22. Sections 02-21 were the train set for the treebank PCFG; only trees with a yield (including punctuation) of length ≤ 20 were used for the pnp program. Coverage in all cases is 1691 (excluding failed parses gives $F_1 = 85.19$ for the tb-pcfg-baseline, and 85.54 for pnp10000).

In figure 8(left) we plot the difference in parse accuracy between the treebank PCFG and our best model per testset sentence. To make the plot more informative, the sentences are ordered by increasing difference. Hence, on the left are sentences where the treebank PCFG scores better, and at the right the sentence where pnp10000 scores best. As is clear from this graph, for most sentences there is no difference, but there are small and about equally sized subsets of sentences for which one or the other model scores better. We have briefly analysed these sentences, but not found a clear pattern. In figure 8(right) we plot in a similar way the difference in log-likelihood that the parsers assign to each sentence. Here we see a clear pattern: only very few sentences receive slightly higher likelihood under the PCFG model. For a good portion of the sentences, however, the pnp10000 model assigns them somewhat and in some cases much higher likelihood. The highest likelihood gains are due to a small number of frequent multiword expressions, such as "New York Stock Exchange Composite Trading", which P-DOP treats as a unit; all of the other gains in likelihood are also due to the use of depth>1 elementary trees, including some non-contiguous constructions such as *revenues rose CD % to $ CD million from $ CD million*.

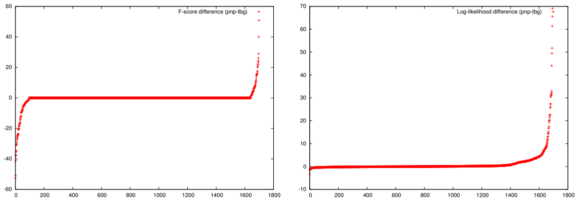


Figure 8: Per sentence difference in f-score (left) and log-likelihood (right) of the sentences of WSJ section 22. The x-axis gives the sentence-rank when sentences are ordered from small to large on y-axis value.

7 Discussion and Conclusions

We set out to develop a parsimonious approach to Data-Oriented Parsing, where all subtrees can potentially become units in a probabilistic grammar but only if the statistics require it. The grammars resulting from our algorithm are orders of magnitude smaller than those used in Bod's maximalist DOP. Although our parsing results are not yet at the level of the best results obtained by Bod, our results indicate that we are getting closer and that we already induce linguistically more plausible grammars.

Could P-DOP eventually not only be more efficient, but also more accurate than maximalist DOP models? Bod has argued that the explanation for DOP's excellent results is that it takes into account all possible dependencies between productions in a tree, and not just those from an a-priori chosen subset (e.g. lexical, head, parent features). Non-head dependencies in non-contiguous natural language constructions, like *more ... than*, as in *more freight than cargo*, are typically excluded in the enrichment/conditioning approaches discussed in section 2. Bod wants to include any dependency a priori, and then "let the statistics decide".

Although the inclusion of all dependencies must somehow explain the performance difference between Bod's best generative model and manually enriched PCFG models, this explanation is not entirely satisfactory. Zuidema (2006a) shows that also the estimator (Bod, 2003) uses is biased and inconsistent, and will, even in the limit of infinite data, not correctly identify many possible distributions over trees. This is not just a theoretical problem. For instance, in the Penn Tree Bank the construction *won't VP* is annotated as *(VP (MD wo) (RB n't) VP)*. There is a strong dependency between the two morphemes: *wo* doesn't exist as an independent word, and strongly predicts *n't*. However, Bod's estimator will continue to reserve probability mass for other combinations with the same POS-tags such as *wo not*, even with an infinite data set only containing *will not* and *wo n't*. Because in parsing the strings are given, this particular example will not harm the parse accuracy results. The example might be diagnostic for other cases that do, however, and certainly will have impact when DOP is used as language model. P-DOP, in contrast, does converge to grammars that treat *won't* as a single unit.

The exact relation of P-DOP to other DOP models, including S-DOP (Bod, 2003), Backoff-DOP (Sima'an and Buratto, 2003), DOP* (Zollmann and Sima'an, 2005) and ML-DOP (Bod, 2006; based on Expectation Maximization) and not dissimilar automatic enrichment models such as (Petrov et al., 2006), remains a topic for future work.

Acknowledgments Funded by the Netherlands Organisation for Scientific Research (EW), project nr. 612.066.405; many thanks to Reut Tsarfaty, Remko Scha, Rens Bod and three anonymous reviewers for their comments.

References

Rens Bod. 1998. *Beyond Grammar: An experience-based theory of language*. CSLI, Stanford, CA.

Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings ACL-2001*.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings EACL'03*.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. *Proceedings ACL-COLING'06*.

Eugene Charniak. 1996. Tree-bank grammars. Technical report, Department of Computer Science, Brown University

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the fourteenth national conference on artificial intelligence*, Menlo Park. AAAI Press/MIT Press.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings ACL 2002*.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings ACL'97*, pages 16–23.

Joshua Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings EMNLP*, pages 143–152.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Mark Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings ACL'03*.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

T. Matzuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings ACL'05*, pages 75–82.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings ACL-COLING'06*, pages 443–440.

Detlef Prescher. 2005. Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *Proceedings ECML'05*.

Remko Scha. 1990. Taaltheorie en taaltechnologie; competence en performance. In R. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, pages 7–22. LVVN, Almere, the Netherlands. English translation at http://iaaa.nl/rs/LeerdamE.html.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings COLING 2004*.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of COLING-ACL 2006*.

Khalil Sima'an. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

Khalil Sima'an and Luciano Buratto. 2003. Backoff parameter estimation for the DOP model. In *Proceedings of the 14th European Conference on Machine Learning (ECML'03, Cavtat-Dubrovnik, Croatia*, number 2837 in Lecture Notes in Artificial Intelligence, pages 373–384. Springer Verlag, Berlin, Germany.

Gert Veldhuijzen van Zanten, Gosse Bouma, Khalil Sima'an, Gertjan van Noord, and Remko Bonnema. 1999. Evaluation of the NLP components of the OVIS2 spoken dialogue system. In van Eynde, Schuurman, and Schelkens, editors, *Computational Linguistics in the Netherlands 1998*, pages 213–229. Rodopi, Amsterdam.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2/3):367–388.

Willem Zuidema. 2006a. Theoretical evaluation of estimation methods for Data-Oriented Parsing. In *Proceedings EACL 2006 (Conference Companion)*, pages 183–186. Association for Computational Linguistics. Erratum on http://staff.science.uva.nl/˜jzuidema/research.

Willem Zuidema. 2006b. What are the productive units of natural language grammar? A DOP approach to the automatic identification of constructions. In *Proceedings of the 10th International Conference on Computational Natural Language Learning (CONLL-X)*.

Generating Lexical Analogies Using Dependency Relations

Andy Chiu, Pascal Poupart, and Chrysanne DiMarco
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
{pachiu,ppoupart,cdimarco}@uwaterloo.ca

Abstract

A lexical analogy is a pair of word-pairs that share a similar semantic relation. Lexical analogies occur frequently in text and are useful in various natural language processing tasks. In this study, we present a system that generates lexical analogies automatically from text data. Our system discovers semantically related pairs of words by using dependency relations, and applies novel machine learning algorithms to match these word-pairs to form lexical analogies. Empirical evaluation shows that our system generates valid lexical analogies with a precision of 70%, and produces quality output although not at the level of the best human-generated lexical analogies.

1 Introduction

Analogy discovery and analogical reasoning are active research areas in a multitude of disciplines, including philosophy, psychology, cognitive science, linguistics, and artificial intelligence. A type of analogy that is of particular interest in natural language processing is *lexical analogy*. A lexical analogy is a pair of word-pairs that share a similar semantic relation. For example, the word-pairs (*dalmatian*, *dog*) and (*trout*, *fish*) form a lexical analogy because dalmatian is a subspecies of dog just as trout is a subspecies of fish, and the word-pairs (*metal*, *electricity*) and (*air*, *sound*) form a lexical analogy because in both cases the initial word serves as a conductor for the second word. Lexical analogies occur frequently in text and are useful in various natural language processing tasks. For example, understanding metaphoric language such as "*the printer died*" requires the recognition of implicit lexical analogies, in this case between (*printer*, *malfunction*) and (*person*, *death*). Lexical analogies also have applications in word sense disambiguation, information extraction, question-answering, and semantic relation classification (see (Turney, 2006)).

In this study, we present a novel system for generating lexical analogies directly from a text corpus without relying on dictionaries or other semantic resources. Our system uses dependency relations to characterize pairs of semantically related words, then compares the similarity of their semantic relations using two machine learning algorithms. We also present an empirical evaluation that shows our system generates valid lexical analogies with a precision of 70%. Section 2 provides a list of definitions, notations, and necessary background materials. Section 3 describes the methods used in our system. Section 4 presents our empirical evaluation. Section 5 reviews selected related work. Finally, Section 6 concludes the paper with suggested future work and a brief conclusion.

2 Definitions

A *word-pair* is a pair of entities, where each entity is a single word or a multi-word named entity. The *underlying relations* of a word-pair (w_1, w_2) are the semantic relations[1] between w_1 and w_2. For exam-

[1] Here 'semantic relations' include both *classical relations* such as synonymy and meronymy, and *non-classical relations* as defined by Morris and Hirst (2004).

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 561–570, Prague, June 2007. ©2007 Association for Computational Linguistics

ple, the underlying relations of (*poet*, *poem*) include *produces*, *writes*, *enjoys*, and *understands*. A *lexical analogy* is a pair of word-pairs that share at least one identical or similar underlying relation.

A key linguistic formalism we use is *dependency grammar* (Tesnière, 1959). A dependency grammar describes the syntactic structure of a sentence in a manner similar to the familiar phrase-structure grammar. However, unlike phrase-structure grammars which associate each word of a sentence to the syntactic phrase in which the word is contained, a dependency grammar associates each word to its syntactic superordinate as determined by a set of rules. Each pair of depending words is called a *dependency*. Within a dependency, the word being depended on is called the *governor*, and the word depending on the governor is called the *dependent*. Each dependency is also labelled with the syntactic relation between the governor and the dependent. Dependency grammars require that each word of a sentence have exactly one governor, except for one word called the *head word* which has no governor at all. A proposition p that is governor to exactly one word w_1 and dependent of exactly one word w_2 is often *collapsed* (Lin and Pantel, 2001); that is, the two dependencies involving p are replaced by a single dependency between w_1 and w_2 labelled p.

The dependency structure of a sentence can be concisely represented by a *dependency tree*, in which each word is a node, each dependent is a child of its governor, and the head word is the root. A *dependency path* is an undirected path through a dependency tree, and a *dependency pattern* is a dependency path with both ends replaced by slots (Lin and Pantel, 2001). Figure 1 illustrates various dependency structures of the sentence, *rebels fired rockets at a military convoy*, after each word is lemmatized.

3 Methods

We consider lexical analogy generation as a sequence of two key problems: *data extraction* and *relation-matching*. Data extraction involves the identification and extraction of pairs of semantically related words, as well as features that characterize their relations. Relation-matching involves matching word-pairs with similar features to form lexical analogies. We describe our methods for solving these two problems in the following subsections.

3.1 Data Extraction

Extracting Word-Pairs

To identify semantically related words, we rely on the assumption that highly syntactically related words also tend to be semantically related — a hypothesis that is supported by works such as Levin's (1993) study of English verbs. As such, the dependency structure of a sentence can be used to approximate the semantic relatedness between its constituent words. Our system uses a dependency parser to parse the input text into a set of dependency trees, then searches through these trees to extract dependency paths satisfying the following constraints:

1. The path must be of the form *noun-verb-noun*.
2. One of the nouns must be the subject of the clause to which it belongs.

Each of these paths is then turned into a word-pair by taking its two nouns. The path constraints that we use are suggested by the *subject-verb-object* (SVO) pattern commonly used in various relation extraction algorithms. However, our constraints allow significantly more flexibility than the SVO pattern in two important aspects. First, our constraints allow an arbitrary relation between the verb and the second noun, not just the object relation. Hence, word-pairs can be formed from a clause's subject and its location, time, instrument, and other arguments, which are clearly semantically related to the subject. Secondly, searching in the space of dependency trees instead of raw text data means that we are able to find semantically related words that are not necessarily adjacent to each other in the sentence.

It is important to note that, although these constraints improve the precision of our system and tend to identify effectively the most relevant word-pairs, they are not strictly necessary. Our system would be fully functional using alternative sets of constraints tailored for specific applications, or even with no constraints at all.

Using the sentence in Figure 1 as an example, our system would extract the dependency paths "*rebel* $\overset{subj}{\leftarrow}$ *fire* $\overset{obj}{\rightarrow}$ *rocket*" and "*rebel* $\overset{subj}{\leftarrow}$ *fire* $\overset{at}{\rightarrow}$ *convoy*", and would thus generate the word-pairs (*rebel*, *rocket*) and (*rebel*, *convoy*).

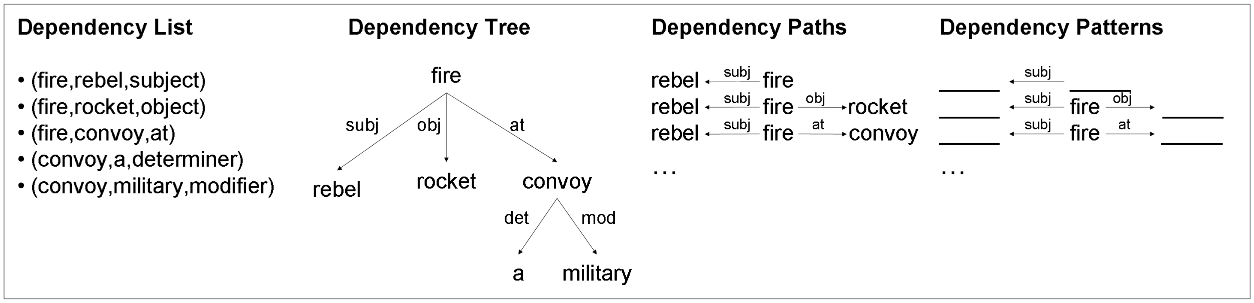


Figure 1: Dependency structures of “*rebels fired rockets at a military convoy*” after lemmatization

Extracting Features

Recall that each word-pair originates from a dependency path. The path, and in particular the middle verb, provides a connection between the two words of the word-pair, and hence is a good indication of their semantic relation. Therefore, for each word-pair extracted, we also extract the dependency pattern derived from the word-pair's dependency path as a feature for the word-pair. We further justify this choice of feature by noting that the use of dependency patterns have previously been shown to be effective at characterizing lexico-syntactic relations (Lin and Pantel, 2001; Snow et al., 2004).

Using Figure 1 as an example again, the dependency patterns “___ $\overset{subj}{\leftarrow}$ *fire* $\overset{obj}{\rightarrow}$ ___” and “___ $\overset{subj}{\leftarrow}$ *fire* $\overset{at}{\rightarrow}$ ___” would be extracted as a feature of (*rebel*, *rocket*) and (*rebel*, *convoy*), respectively.

Filtering

Word-pairs and features extracted using only dependency relations tend to be crude in several aspects. First, they contain a significant amount of noise, such as word-pairs that have no meaningful underlying relations. Noise comes from grammatical and spelling mistakes in the original input data, imperfect parsing, as well as the fact that dependency structure only approximates semantic relatedness. Secondly, some of the extracted word-pairs contain underlying relations that are too general or too obscure for the purpose of lexical analogy generation. For example, consider the word-pair (*company*, *right*) from the sentence “*the company exercised the right to terminate his contract*”. The two words are clearly semantically related, however the relation (*have* or *entitled-to*) is very general and it is difficult to construct satisfying lexical analogies from the word-pair. Lastly, some features are also subject to the same problem. The feature “___ $\overset{subj}{\leftarrow}$ *say* $\overset{obj}{\rightarrow}$ ___”, for example, has very little characterization power because almost any pair of words can occur with this feature.

In order to retain only the most relevant word-pairs and features, we employ a series of refining filters. All of our filters rely on the occurrence statistics of the word-pairs and features. Let $\mathcal{W} = \{wp_1, wp_2, ..., wp_n\}$ be the set of all word-pairs and $\mathcal{F} = \{f_1, f_2, ..., f_m\}$ the set of all features. Let $\mathcal{F}_{wp}$ be the set of features of word-pair wp, and let $\mathcal{W}_f$ be the set of word-pairs associated with feature f. Let $O(wp)$ be the total number of occurrences of word-pair wp, $O(f)$ be the total number of occurrences of feature f, and $O(wp, f)$ be the number of occurrences of word-pair wp with feature f. The following filters are used:

1. Occurrence filter: Eliminate word-pair wp if $O(wp)$ is less than some constant K_{f_1}, and eliminate feature f if $O(f)$ is less than some constant K_{f_2}. This filter is inspired by the simple observation that valid word-pairs and features tend to occur repeatedly.

2. Generalization filter: Eliminate feature f if $|\mathcal{W}_f|$ is greater than some constant K_{f_3}. This filter ensures that features associated with too many word-pairs are not kept. A feature that occurs with many word-pairs tend to describe overly general relations. An example of such a feature is “___ $\overset{subj}{\leftarrow}$ *say* $\overset{obj}{\rightarrow}$ ___”, which in our experiment occurred with several thousand word-pairs while most features occurred with less than a hundred.

3. Data sufficiency filter: Eliminate word-pair wp if $|\mathcal{F}_{wp}|$ is less than some constant K_{f_4}. This filter ensures that all word-pairs have sufficient features to be compared meaningfully.

4. Entropy filter: Eliminate word-pair wp if its *normalized entropy* is greater than some constant K_{f_5}. We compute a word-pair's entropy by considering it as a distribution over features, in a manner that is analogous to the *feature entropy* defined in (Turney, 2006). Specifically, the normalized entropy of a word-pair wp is:

$$-\frac{\sum_{f \in \mathcal{F}_{wp}} p(f|wp) \log \left(p(f|wp)\right)}{\log |\mathcal{F}_{wp}|}$$

   where $p(f|wp) = \frac{O(wp,f)}{O(wp)}$ is the conditional probability of f occurring in the context of wp. The normalized entropy of a word-pair ranges from zero to one, and is at its highest when the distribution of the word-pair's occurrences over its features is the most random. The justification behind this filter is that word-pairs with strong underlying relations tend to have just a few dominant features that characterize those relations, whereas word-pairs that have many non-dominant features tend to have overly general underlying relations that can be characterized in many different ways.

3.2 Relation-Matching

Central to the problem of relation-matching is that of a *relational similarity function*: a function that computes the degree of similarity between two word-pairs' underlying relations. Given such a function, relation-matching reduces to simply computing the relational similarity between every pair of word-pairs, and outputting the pairs scoring higher than some threshold K_{th} as lexical analogies. Our system incorporates two relational similarity functions, as discussed in the following subsections.

Latent Relational Analysis

The baseline algorithm that we use to compute relational similarity is a modified version of Latent Relational Analysis (LRA) (Turney, 2006), that consists of the following steps:

1. Construct an n-by-m matrix A such that the ith row maps to word-pair wp_i, the jth column maps to feature f_j, and $A_{i,j} = O(wp_i, f_j)$.

2. Reduce the dimensionality of A to a constant K_{svd} using Singular Value Decomposition (SVD) (Golub and van Loan, 1996). SVD produces a matrix $\hat{A}$ of rank K_{svd} that is the best approximation of A among all matrices of rank K_{svd}. The use of SVD to compress the feature space was pioneered in Latent Semantic Analysis (Deerwester et al., 1990) and has become a popular technique in feature-based similarity computation. The compressed space is believed to be a semantic space that minimizes artificial surface differences.

3. The relational similarity between two word-pairs is the cosine measure of their corresponding row vectors in the reduced feature space. Specifically, let $\hat{A}_i$ denote the ith row vector of $\hat{A}$, then the relational similarity between word-pairs wp_{i_1} and wp_{i_2} is:

$$\frac{\hat{A}_{i_1} \cdot \hat{A}_{i_2}}{\left\|\hat{A}_{i_1}\right\|_2 + \left\|\hat{A}_{i_2}\right\|_2}$$

The primary difference between our algorithm and LRA is that LRA also includes each word's synonyms in the computation. Synonym inclusion greatly increases the size of the problem space, which leads to computational issues for our system as it operates at a much larger scale than previous work in relational similarity. Turney's (2006) extensive evaluation of LRA on SAT verbal analogy questions, for example, involves roughly ten thousand relational similarity computations[2]. In contrast, our system typically requires millions of relational similarity computations because every pair of extracted word-pairs needs to be compared. We call our algorithm LRA-S (LRA Without Synonyms) to differentiate it from the original LRA.

Similarity Graph Traversal

While LRA has been shown to perform well in computing relational similarity, it suffers from two

[2] The study evaluated 374 SAT questions, each involving 30 pairwise comparisons, for a total of 11220 relational similarity computations.

limitations. First, the use of SVD is difficult to interpret from an analytical point of view as there is no formal analysis demonstrating that the compressed space really corresponds to a semantic space. Secondly, even LRA-S does not scale up well to large data sets due to SVD being an expensive operation — computing SVD is in general $O(mn \cdot \min(m, n))$ (Koyuturk et al., 2005), where m, n are the number of matrix rows and columns, respectively.

To counter these limitations, we propose an alternative algorithm for computing relational similarity — Similarity Graph Traversal (SGT). The intuition behind SGT is as follows. Suppose we know that wp_1 and wp_2 are relationally similar, and that wp_2 and wp_3 are relationally similar. Then, by transitivity, wp_1 and wp_3 are also likely to be relationally similar. In other words, the relational similarity between two word-pairs can be reinforced by other word-pairs through transitivity. The actual algorithm involves the following steps:

1. Construct a *similarity graph* as follows. Each word-pair corresponds to a node in the graph. An edge exists from wp_1 to wp_2 if and only if the cosine measure of the two word-pairs' feature vectors is greater than or equal to some threshold K_{sgt}, in which case, the cosine measure is assigned as the *strength* of the edge.

2. Define a *similarity path* of length k, or k-*path*, from wp_1 to wp_2 to be a directed acyclic path of length k from wp_1 to wp_2, and define the *strength* $s(p)$ of a path p to be the product of the strength of all of the path's edges. Denote the set of all k-paths from wp_1 to wp_2 as $\mathcal{P}(k, wp_1, wp_2)$, and denote the sum of the strength of all paths in $\mathcal{P}(k, wp_1, wp_2)$ as $\mathcal{S}(k, wp_1, wp_2)$.

3. The relational similarity between word-pairs wp_{i_1} and wp_{i_2} is:

$$\begin{gathered} \alpha_1 \mathcal{S}(1, wp_1, wp_2) + \\ \alpha_2 \mathcal{S}(2, wp_1, wp_2) + \\ \cdots \\ \alpha_{K_l} \mathcal{S}(K_l, wp_1, wp_2) \end{gathered}$$

   where K_l is the maximum path length to consider, and α_1, …, α_{K_l} are weights that are learned using least-squares regression on a small set of hand-labelled lexical analogies.

A natural concern for SGT is that relational similarity is not always transitive, and hence some paths may be invalid. For example, although (*teacher*, *student*) is relationally similar to both (*shepherd*, *sheep*) and (*boss*, *employee*), the latter two word-pairs are not relationally similar. The reason that this is not a problem for SGT is because truly similar word-pairs tend to be connected by many transitive paths, while invalid paths tend to occur in isolation. As such, while a single path may not be indicative, a collection of many paths likely signifies a true common relation. The weights in step 3 ensure that SGT assigns a high similarity score to two word-pairs only if there are sufficiently many transitive paths (which are sufficiently strong) between them.

Analogy Filters

As a final step in both LSA-R and SGT, we filter out lexical analogies of the form (w_1,w_2) and (w_1,w_3), as such lexical analogies tend to express the near-synonymy between w_2 and w_3 more than they express the relational similarity between the two word-pairs. We also keep only one *permutation* of each lexical analogy: (w_1,w_2) and (w_3,w_4), (w_3,w_4) and (w_1,w_2), (w_2,w_1) and (w_4,w_3), and (w_4,w_3) and (w_2,w_1) are different permutations of the same lexical analogy.

4 Evaluation

Our evaluation consisted of two parts. First, we evaluated the performance of the system, using LRA-S for relation-matching. Then, we evaluated the SGT algorithm, in particular, how it compares to LRA-S.

4.1 System Evaluation

Experimental Setup

We implemented our system in Sun JDK 1.5. We also used MXTerminator (Reynar and Ratnaparkhi, 1997) for sentence segmentation, MINIPAR (Lin, 1993) for lemmatization and dependency parsing, and MATLAB[3] for SVD computation. The experiment was conducted on a 2.1 GHz processor, with

[3] http://www.mathworks.com

the exception of SVD computation which was carried out in MATLAB running on a single 2.4 GHz processor within a 64-processor cluster. The input corpus consisted of the following collections in the Text Retrieval Conference Dataset[4]: AP Newswire 1988–1990, LA Times 1989–1990, and San Jose Mercury 1991. In total, 1196 megabytes of text data were used for the experiment. Table 1 summarizes the running times of the experiment.

Process	Time
Sentence Segmentation	20 min
Dependency Parsing	2232 min
Data Extraction	138 min
Relation-Matching	65 min

Table 1: Experiment Running Times

The parameter values selected for the experiment are listed in Table 2. The filter parameters were selected mostly through trial-and-error — various parameter values were tried and filtration results examined. We used a threshold value K_{th} = 0.80 to generate the lexical analogies, but the evaluation was performed at ten different thresholds from 0.98 to 0.80 in 0.02 decrements.

K_{f_1}	K_{f_2}	K_{f_3}	K_{f_4}	K_{f_5}	K_{svd}
35	10	100	10	0.995	600

Table 2: Experiment Parameter Values

Evaluation Protocol

An objective evaluation of our system is difficult for two reasons. First, lexical analogies are by definition subjective; what constitutes a 'good' lexical analogy is debatable. Secondly, there is no gold standard of lexical analogies to which we can compare. For these reasons, we adopted a subjective evaluation protocol that involved human judges rating the quality of the lexical analogies generated. Such a manual evaluation protocol, however, meant that it was impractical to evaluate the entire output set (which was well in the thousands). Instead, we evaluated random samples from the output and interpolated the results.

In total, 22 human judges participated in the evaluation. All judges were graduate or senior undergraduate students in English, Sociology, or Psychology, and all were highly competent English speakers. Each judge was given a survey containing 105 lexical analogies, 100 of which were randomly sampled from our output, and the remaining five were sampled from a control set of ten human-generated lexical analogies. All entries in the control set were taken from the Verbal Analogy section of the Standard Aptitude Test[5] and represented the best possible lexical analogies. The judges were instructed to grade each lexical analogy with a score from zero to 10, with zero representing an invalid lexical analogy (i.e., when the two word-pairs share no meaningful underlying relation) and ten representing a perfect lexical analogy. To minimize inter-judge subjectivity, all judges were given detailed instructions containing the definition and examples of lexical analogies. In all, 1000 samples out of the 8373 generated were graded, each by at least two different judges.

We evaluated the output at ten threshold values, from 0.98 to 0.80 in 0.02 decrements. For each threshold, we collected all samples down to that threshold and computed the following metrics:

1. Coverage: The number of lexical analogies generated at the current threshold over the number of lexical analogies generated at the lowest threshold (8373).

2. Precision: The proportion of samples at the current threshold that scored higher than three. These are considered valid lexical analogies. Note that this is significantly more conservative than the survey scoring. We want to ensure very poor lexical analogies were excluded, even if they were 'valid' according to the judges.

3. Quality: The average score of all samples at the current threshold, divided by ten to be in the same scale as the other metrics.

4. Goodness: The proportion of samples at the current threshold that scored within 10% of the average score of the control set. These are considered human quality.

[4] http://trec.nist.gov/

[5] http://www.collegeboard.com/

Note that recall was not an evaluation metric because there does not exist a method to determine the true number of lexical analogies in the input corpus.

Result

Table 3 summarizes the result of the control set, and Figure 2:Left summarizes the result of the lexical analogies our system generated. Table 4 lists some good and some poor lexical analogies our system generated, along with some of their shared features.

Coverage	Precision	Quality	Goodness
N/A	1.00	0.97	0.90

Table 3: Result of the Control Set

As Figure 2 shows, our system performed fairly well, generating valid lexical analogies with a precision around 70%. The quality of the generated lexical analogies was reasonable, although not at the level of human-generation. On the other hand, a small portion (19% at the highest threshold) of our output was of very high quality, comparable to the best human-generated lexical analogies.

Our result also showed that there was a correspondence between the score our system assigned to each generated lexical analogy and its quality. Precision, quality, and goodness all declined steadily toward lower thresholds: precision 0.70–0.66, quality 0.54–0.49, and goodness 0.19–0.14.

Error Analysis

Despite our aggressive filtration of irrelevant word-pairs and features, noise was still the most significant problem in our output. Most low-scoring samples contained at least one word-pair that did not have a meaningful and clear underlying relation; for examples, (*guy*, *ball*) and (*issue*, *point*). As mentioned, noise originated from mistakes in the input data, errors in sentence segmentation and parsing, as well as mismatches between dependencies and semantic relatedness. An example of the latter involved the frequent usage of the proposition "*of*" in various constructs. In the sentence "*the company takes advantage of the new legislation*", for example, the dependency structure associates *company* with *advantage*, whereas the semantic relation clearly lies between *company* and *legislation*. All three of our evaluation metrics (precision, quality, and goodness) were negatively affected by noise.

Polysemic words, as well as words which were heavily context-dependent, also posed a problem. For example, one of the lexical analogies generated in the experiment was (*resolution*, *house*) and (*legislation*, *senate*). This lexical analogy only makes sense if "*house*" is recognized as referring to the House of Representatives, which is often abbreviated as "*the House*" in news articles. Polysemy also negatively affected all three of our evaluation metrics, although to a lesser extent for precision.

Finally, our system had difficulties differentiating semantic relations of different granularity. The underlying relations of (*relation*, *country*) and (*tie*, *united states*), for example, are similar, yet they do not form a good lexical analogy because the relations are at different levels of granularity (countries in general in the former, and a particular country in the latter). Undifferentiated granularity affected quality and goodness, but it did not have a significant effect on precision.

4.2 SGT Evaluation

To evaluate how SGT compares to LRA-S, we repeated the experiment using SGT for relation-matching. We set K_l (maximum path length) to 3, and K_{sgt} (cosine threshold) to 0.2; these values were again determined largely through trial-and-error. To train SGT, we used 90 lexical analogies graded by human judges from the previous experiment. In order to facilitate a fair comparison to LRA-S, we selected K_{th} values that allowed SGT to generate the same number of lexical analogies as LRA-S did at each threshold interval.

Running on the same 2.1 GHz processor, SGT finished in just over eight minutes, which is almost a magnitude faster than LRA-S' 65 minutes. SGT also used significantly less memory, as the similarity graph was efficiently stored in an adjacency list. The sets of lexical analogies generated by the two algorithms were quite similar, overlapping approximately 50% at all threshold levels.

The significant overlap between SGT and LRA-S' outputs allowed us to evaluate SGT using the samples collected from the previous surveys instead of conducting a new round of human grading. Specifically, we identified previously graded samples that

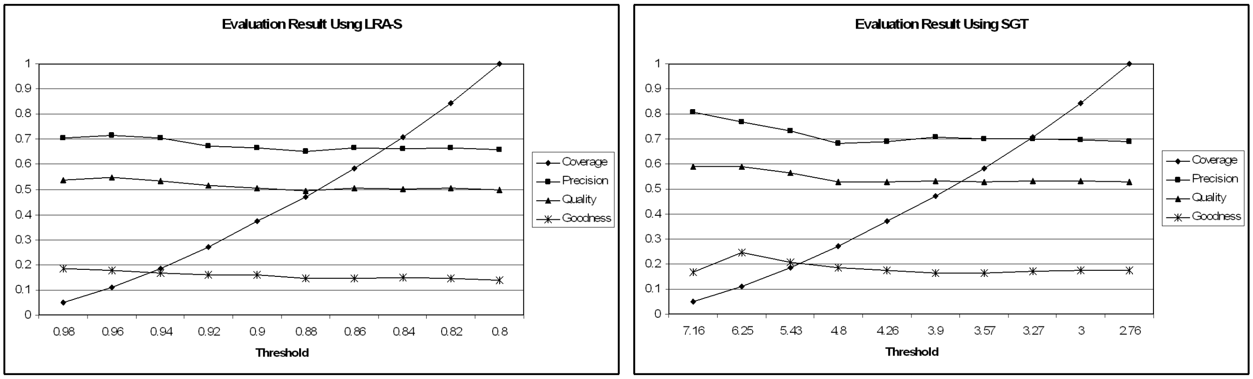


Figure 2: System Evaluation Results

Good Examples	Shared Features
(*vietnam, cambodia*) and (*iraq, kuwait*)	___ $\xleftarrow{subj}$ *invade* $\xrightarrow{obj}$ ___, ___ $\xleftarrow{subj}$ *pull out* $\xrightarrow{of}$ ___
(*building, office*) and (*museum, collection*)	___ $\xleftarrow{subj}$ *house* $\xrightarrow{obj}$ ___, ___ $\xleftarrow{subj}$ *consolidate* $\xrightarrow{obj}$ ___
(*stock market, rally*) and (*student, march*)	___ $\xleftarrow{subj}$ *stage* $\xrightarrow{obj}$ ___
(*researcher, experiment*) and (*doctor, surgery*)	___ $\xleftarrow{subj}$ *perform* $\xrightarrow{obj}$ ___
(*gainer, loser*) and (*decline, advance*)	___ $\xleftarrow{subj}$ *outnumber* $\xrightarrow{obj}$ ___
(*book, shelf*) and (*picture, wall*)	___ $\xleftarrow{with}$ *line* $\xrightarrow{subj}$ ___, ___ $\xleftarrow{subj}$ *remain* $\xrightarrow{on}$ ___
(*blast, car*) and (*sanction, economy*)	___ $\xleftarrow{subj}$ *damage* $\xrightarrow{obj}$ ___, ___ $\xleftarrow{by}$ *destroy* $\xrightarrow{subj}$ ___
Poor Examples	Shared Features
(*president, change*) and (*bush, legislation*)	___ $\xleftarrow{subj}$ *veto* $\xrightarrow{obj}$ ___
(*charge, death*) and (*lawsuit, federal court*)	___ $\xleftarrow{subj}$ *file* $\xrightarrow{in}$ ___
(*relation, country*) and (*tie, united states*)	___ $\xleftarrow{obj}$ *severe* $\xrightarrow{subj}$ ___
(*judge, term*) and (*member, life*)	___ $\xleftarrow{subj}$ *sentence* $\xrightarrow{to}$ ___
(*issue, point*) and (*stock, cent*)	___ $\xleftarrow{subj}$ *be* $\xrightarrow{down}$ ___, ___ $\xleftarrow{subj}$ *be* $\xrightarrow{up}$ ___

Table 4: Examples of Good and Poor Lexical Analogies Generated

had also been generated by SGT, and used these samples as the evaluation data points for SGT. At the lowest threshold (where 8373 lexical analogies were generated), we were able to reuse 533 samples out of the original 1000 samples. Figure 2:Right summarizes the performance of the system using SGT for relation-matching.

As the figure shows, SGT performed very similarly to LRA-S. Both SGT's precision and quality scores were slightly higher than LRA-S, but the differences were very small and hence were likely due to sample variation. The goodness scores between the two algorithms were also comparable. In the case of SGT, however, the score fluctuated instead of monotonically decreased. We attribute the fluctuation to the smaller sample size.

As the samples were drawn exclusively from the portion of SGT's output that overlapped with LRA-S' output, we needed to ensure that the samples were not strongly biased and that the reported result was not better than SGT's actual performance. To validate the result, we conducted an additional experiment involving a single human judge. The judge was given a survey with 50 lexical analogies, 25 of which were sampled from the overlapping portion of SGT and LRA-S' outputs, and 25 from lexical analogies generated only by SGT. Table 5 summarizes the result of this experiment. As the table demonstrates, the results from the two sets were comparable with small differences. Moreover, the differences were in favour of the SGT-only portion. Therefore, either there was no sampling bias at all, or the sampling bias negatively affected the result. As such, SGT's actual performance was at least as good as reported, and may have been slightly higher.

	Precision	Quality	Goodness
Overlap	0.76	0.56	0.28
SGT-Only	0.88	0.62	0.2

Table 5: Overlap vs. SGT-Only

We conclude that SGT is indeed a viable alternative to LRA-S. SGT generates lexical analogies that are of the same quality as LRA-S, while being significantly faster and more scalable. On the other hand, an obvious limitation of SGT is that it is a supervised algorithm requiring manually labelled training data. We claim this is not a severe limitation because there are only a few variables to train (i.e., the weights), hence only a small set of training data is required. Moreover, a supervised algorithm can be advantageous in some situations; for example, it is easier to tailor SGT to a particular input corpus.

5 Related Work

The study of analogy in the artificial intelligence community has historically focused on computational models of analogy-making. French (2002) and Hall (1989) provide two of the most complete surveys of such models. Veale (2004; 2005) generates lexical analogies from WordNet (Fellbaum, 1998) and HowNet (Dong, 1988) by dynamically creating new type hierarchies from the semantic information stored in these lexicons. Unlike our corpus-based generation system, Veale's algorithms are limited by the lexicons in which they operate, and generally are only able to generate *near-analogies* such as (*Christian*, *Bible*) and (*Muslim*, *Koran*). Turney's (2006) Latent Relational Analysis is a corpus-based algorithm that computes the relational similarity between word-pairs with remarkably high accuracy. However, LRA is focused solely on the relation-matching problem, and by itself is insufficient for lexical analogy generation.

6 Conclusion and Future Work

We have presented a system that is, to the best of our knowledge, the first system capable of generating lexical analogies from unstructured text data. Empirical evaluation shows that our system performed fairly well, generating valid lexical analogies with a precision of about 70%. The quality of the generated lexical analogies was reasonable, although not at the level of human performance. As part of the system, we have also developed a novel algorithm for computing relational similarity that rivals the performance of the current state-of-the-art while being significantly faster and more scalable. One of our immediate tasks is to complement dependency patterns with additional features. In particular, we expect semantic features such as word definitions from machine-readable dictionaries to improve our system's ability to differentiate between different senses of polysemic words, as well as different granularities of semantic relations. We also plan to take advantage of our system's flexibility and relax the constraints on dependency paths so as to generate more-varied lexical analogies, e.g., analogies involving verbs and adjectives.

A potential application of our system, and the original inspiration for this research, would be to use the system to automatically enrich ontologies by spreading semantic relations between lexical analogues. For example, if words w_1 and w_2 are related by relation r, and (w_1, w_2) and (w_3, w_4) form a lexical analogy, then it is likely that w_3 and w_4 are also related by r. A dictionary of lexical analogies therefore would allow an ontology to grow from a small set of seed relations. In this way, lexical analogies become bridges through which semantic relations flow in a sea of ontological concepts.

Acknowledgments

We thank the reviewers of EMNLP 2007 for valuable comments and suggestions. This work was supported in part by the Ontario Graduate Scholarship Program, Ontario Innovation Trust, Canada Foundation for Innovation, and the Natural Science and Engineering Research Council of Canada.

References

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Dong Zhen Dong. 1988. What, how and who? *Proceedings of the International Symposium on Electronic Dictionaries*. Tokyo, Japan.

Christine Fellbaum, editor. 1998. *WordNet — An Electronic Lexical Database*. MIT Press.

Robert French. 2002. The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200–205.

Gene Golub and Charles van Loan. 1996. *Matrix Computations*. Johns Hopkins University Press, third edition.

Rogers Hall. 1989. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39:39–120.

Mehmet Koyuturk, Ananth Grama, and Naren Ramakrishnan. 2005. Compression, clustering, and pattern discovery in very high-dimensional discrete-attribute data sets. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):447–461.

Beth Levin 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Dekang Lin. 1993. Principle-based parsing without overgeneration. *Proceedings of the 31st Annual Meeting on ACL*, pp 112–120. Columbus, USA.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Jane Morris and Graeme Hirst. 2004. Non-classical lexical semantic relations. *Proceedings of the Computational Lexical Semantics Workshop at HLT-NAACL 2004*, pp 46–51. Boston, USA.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp 16–19. Washington, USA.

Gerard Salton, A. Wong, and C.S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 13(11):613–620.

Rion Snow, Daniel Jurafsky, and Andrew Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Proceedings of the 2004 Neural Information Processing Systems Conference*. Vancouver, Canada.

Lucien Tesnière. 1959. *Éléments de Syntaxe Structurale*. Librairie C. Klincksieck, Paris.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Tony Veale, Jer Hayes, and Nuno Seco. 2004. The Bible is the Christian Koran: Discovering simple analogical compounds. *Proceedings of the Workshop on Computational Creativity in 2004 European Conference on Case-Based Reasoning*. Madrid, Spain.

Tony Veale. 2005. Analogy generation with HowNet. *Proceedings of the 2005 International Joint Conference on Artificial Intelligence*. Edinburgh, Scotland.

Cross-lingual Distributional Profiles of Concepts for Measuring Semantic Distance

Saif Mohammad[†] **Iryna Gurevych**[*] **Graeme Hirst**[†] **Torsten Zesch**[*]

[†]Dept. of Computer Science
University of Toronto
Toronto, Canada
{smm,gh}@cs.toronto.edu

[*]Ubiquitous Knowledge Processing Group
Darmstadt University of Technology
Darmstadt, Germany
{gurevych,zesch}@tk.informatik.tu-darmstadt.de

Abstract

We present the idea of estimating semantic distance in one, possibly resource-poor, language using a knowledge source in another, possibly resource-rich, language. We do so by creating cross-lingual distributional profiles of concepts, using a bilingual lexicon and a bootstrapping algorithm, but without the use of any sense-annotated data or word-aligned corpora. The cross-lingual measures of semantic distance are evaluated on two tasks: (1) estimating semantic distance between words and ranking the word pairs according to semantic distance, and (2) solving *Reader's Digest* 'Word Power' problems. In task (1), cross-lingual measures are superior to conventional monolingual measures based on a wordnet. In task (2), cross-lingual measures are able to solve more problems correctly, and despite scores being affected by many tied answers, their overall performance is again better than the best monolingual measures.

1 Introduction

Accurately estimating the semantic distance between concepts or between words in context has pervasive applications in computational linguistics, including machine translation, information retrieval, speech recognition, spelling correction, and text categorization (see Budanitsky and Hirst (2006) for discussion), and it is becoming clear that basing such measures on a combination of corpus statistics with a knowledge source, such as a dictionary, published thesaurus, or WordNet, can result in higher accuracies (Mohammad and Hirst, 2006b). This is because such knowledge sources capture semantic information about concepts and, to some extent, world knowledge. They also act as sense inventories for the words in a language.

However, applying algorithms for semantic distance to most languages is hindered by the lack of linguistic resources. In this paper, we propose a new method that allows us to compute semantic distance in a possibly resource-poor language by seamlessly combining its text with a knowledge source in a different, preferably resource-rich, language. We demonstrate the approach by combining German text with an English thesaurus to create English–German distributional profiles of concepts, which in turn will be used to measure the semantic distance between German words.

Two classes of methods have been used in determining semantic distance. **Semantic measures of concept-distance**, such as those of Jiang and Conrath (1997) and Resnik (1995), rely on the structure of a knowledge source, such as WordNet, to determine the distance between two concepts defined in it (see Budanitsky and Hirst (2006) for a survey). **Distributional measures of word-distance**[1], such as cosine and α-skew divergence (Lee, 2001), deem

[1] Many distributional approaches represent the sets of contexts of the target words as points in multidimensional co-occurrence space or as co-occurrence distributions. A measure, such as cosine, that captures vector distance or a measure, such as α-skew divergence, that captures distance between distributions is then used to measure distributional distance. We will therefore refer to these measures as distributional measures.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 571–580, Prague, June 2007. ©2007 Association for Computational Linguistics

two words to be closer or less distant if they occur in similar contexts (see Mohammad and Hirst (2005) for a comprehensive survey).

Distributional measures rely simply on raw text and possibly some shallow syntactic processing. They do not require any other manually-created resource, and tend to have a higher coverage. However, by themselves they perform poorly when compared to semantic measures (Mohammad and Hirst, 2006b) because when given a target word pair we usually need the distance between their closest senses, but distributional measures of word-distance tend to conflate the distances between all possible sense pairs. Latent semantic analysis (LSA) (Landauer et al., 1998) has also been used to measure distributional distance with encouraging results (Rapp, 2003). However, it too measures the distance between words and not senses. Further, the dimensionality reduction inherent to LSA has the effect of making the predominant sense more dominant while de-emphasizing the other senses. Therefore, an LSA-based approach will also conflate information from the different senses, and even more emphasis will be placed on the predominant senses. Given the semantically close target nouns *play* and *actor*, for example, a distributional measure will give a score that is some sort of a dominance-based average of the distances between their senses. The noun *play* has the predominant sense of 'children's recreation' (and not 'drama'), so a distributional measure will tend to give the target pair a large (and thus erroneous) distance score. Also, distributional word-distance approaches need to create large $V \times V$ co-occurrence and distance matrices, where V is the size of the vocabulary (usually at least 100,000).[2]

Mohammad and Hirst (2006b) proposed a way of combining written text with a published thesaurus to measure distance between *concepts* (or word senses) using distributional measures, thereby eliminating sense-conflation and achieving results better than the simple word-distance measures and indeed also most of the WordNet-based semantic measures. We called these measures **distributional measures of concept-distance**. Concept-distance measures can be used to measure distance between a word pair by choosing the distance between their closest senses. Thus, even though 'children's recreation' is the predominant sense of *play*, the 'drama' sense is much closer to *actor* and so their distance will be chosen. These distributional concept-distance approaches need to create only $V \times C$ co-occurrence and $C \times C$ distance matrices, where C is the number of categories or senses (usually about 1000). It should also be noted that unlike the best WordNet-based measures, distributional measures (both word- and concept-distance ones) can be used to estimate not just semantic similarity but also semantic relatedness—useful in many tasks including information retrieval. However, the high-quality thesauri and (to a much greater extent) WordNet-like resources that these methods require do not exist for most of the 3000–6000 languages in existence today and they are costly to create.

In this paper, we introduce **cross-lingual distributional measures of concept-distance**, or simply **cross-lingual measures**, that determine the distance between a word pair belonging to a resource-poor language using a knowledge source in a resource-rich language and a bilingual lexicon[3]. We will use the cross-lingual measures to calculate distances between German words using an English thesaurus and a German corpus. Although German is not resource-poor *per se*, Gurevych (2005) has observed that the German wordnet GermaNet (Kunze, 2004) (about 60,000 synsets) is less developed than the English WordNet (Fellbaum, 1998) (about 117,000 synsets) with respect to the coverage of lexical items and lexical semantic relations represented therein. On the other hand, substantial raw corpora are available for the German language. Crucially for our evaluation, the existence of GermaNet allows comparison of our cross-lingual approach with monolingual ones.

2 Monolingual Distributional Measures

In order to set the context for cross-lingual concept-distance measures (Section 3), we first summarize monolingual distributional approaches, with a focus on distributional concept-distance measures.

[2] LSA is especially expensive as singular value decomposition, a key component for dimensionality reduction, requires computationally intensive matrix operations; making it less scalable to large amounts of text (Gorman and Curran, 2006).

[3] For most languages that have been the subject of academic study, there exists at least a bilingual lexicon mapping the core vocabulary of that language to a major world language and a corpus of at least a modest size.

2.1 Word-distance

Words that occur in similar contexts tend to be semantically close. In our experiments, we defined the context of a target word, its co-occurring words, to be ±5 words on either side (but not crossing sentence boundaries). The set of contexts of a target word is usually represented by the strengths of association of the target with its co-occurring words, which we refer to as the **distributional profile (DP)** of the word. Here is a constructed example DP of the word *star*:

DP of a word
star: *space* 0.28, *movie* 0.2, *famous* 0.13,
light 0.09, *rich* 0.04, ...

Simple counts are made of how often the target word co-occurs with other words in text and how often the words occur individually. A suitable statistic, such as pointwise mutual information (PMI), is then applied to these counts to determine the strengths of association between the target and co-occurring words. The distributional profiles of two target words represent their contexts as points in multi-dimensional word-space. A suitable distributional measure (for example, cosine) gives the distance between the two points, and thereby an estimate of the semantic distance between the target words.

2.2 Concept-distance

In Mohammad and Hirst (2006b), we show how distributional profiles of *concepts* (DPCs) can be used to measure semantic distance. Below are the DPCs or DPs of two senses of the word *star* (the senses or concepts themselves are glossed by a set of near-synonymous words, placed in parentheses):

DPs of concepts
'celestial body' (*celestial body, sun, ...*): *space* 0.36, *light* 0.27,
constellation 0.11, ...
'celebrity' (*celebrity, hero, ...*):
famous 0.24, *movie* 0.14, *rich* 0.14, ...

Thus the profiles of two target *concepts* represent their contexts as points in multi-dimensional word-space. A suitable distributional measure (for example, cosine) can then be used to give the distributional distance between the two concepts in the same way that distributional word-distance is measured.

But to calculate the strength of association of a concept with co-occurring words, in order to create DPCs, we must determine the number of times a word used in that sense co-occurs with surrounding words. In Mohammad and Hirst (2006a), we proposed a way to determine these counts without the use of sense-annotated data. Briefly, a **word–category co-occurrence matrix (WCCM)** is created having English word types w^{en} as one dimension and English thesaurus categories c^{en} as another. We used the *Macquarie Thesaurus* (Bernard, 1986) both as a very coarse-grained sense inventory and a source of possibly ambiguous English words that together unambiguously represent each category (concept). The WCCM is populated with co-occurrence counts from a large English corpus (we used the *British National Corpus (BNC)*). A particular cell m_{ij}, corresponding to word w_i^{en} and concept c_j^{en}, is populated with the number of times w_i^{en} co-occurs (in a window of ±5 words) with any word that has c_j^{en} as one of its senses (i.e., w_i^{en} co-occurs with any word listed under concept c_j^{en} in the thesaurus).

$$\begin{array}{c|cccccc}
 & c_1^{en} & c_2^{en} & \dots & c_j^{en} & \dots \\
\hline
w_1^{en} & m_{11} & m_{12} & \dots & m_{1j} & \dots \\
w_2^{en} & m_{21} & m_{22} & \dots & m_{2j} & \dots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
w_i^{en} & m_{i1} & m_{i2} & \dots & m_{ij} & \dots \\
\vdots & \vdots & \vdots & \dots & \vdots & \ddots
\end{array}$$

This matrix, created after a first pass of the corpus, is the **base word–category co-occurrence matrix (base WCCM)** and it captures strong associations between a sense and co-occurring words.[4] This is similar to how Yarowsky (1992) identifies words that are indicative of a particular sense of the target.

We know that words that occur close to a target word tend to be good indicators of its intended sense. Therefore, we make a second pass of the corpus, using the base WCCM to roughly disambiguate the words in it. For each word, the strength of association of each of the words in its context (±5 words)

[4] From the base WCCM we can determine the number of times a word w and concept c co-occur, the number of times w co-occurs with any concept, and the number of times c co-occurs with any word. A statistic such as PMI can then give the strength of association between w and c.

with each of its senses is summed. The sense that has the highest cumulative association is chosen as the intended sense. A new **bootstrapped WCCM** is created such that each cell m_{ij}, corresponding to word w_i^{en} and concept c_j^{en}, is populated with the number of times w_i^{en} co-occurs with any word *used in sense* c_j^{en}.

Mohammad and Hirst (2006a) used the DPCs created from the bootstrapped WCCM to attain near-upper-bound results in the task of determining word sense dominance. Unlike the McCarthy et al. (2004) dominance system, our approach can be applied to much smaller target texts (a few hundred sentences) without the need for a large similarly-sense-distributed text[5]. In Mohammad and Hirst (2006a), the DPC-based monolingual distributional measures of *concept-distance* were used to rank word pairs by their semantic similarity and to correct real-word spelling errors, attaining markedly better results than monolingual distributional measures of *word-distance*. In the spelling correction task, the distributional concept-distance measures performed better than all WordNet-based measures as well, except for the Jiang and Conrath (1997) measure.

3 Cross-lingual Distributional Measures

We now describe how distributional measures of concept-distance can be used in a cross-lingual framework to determine the distance between words in (resource-poor) language L_1 by combining its text with a thesaurus in (resource-rich) language L_2, using an L_1–L_2 bilingual lexicon. We will compare this approach with the best monolingual approaches; the smaller the loss in performance, the more capable the algorithm is of overcoming ambiguities in word translation. An evaluation, therefore, requires an L_1 that in actuality has adequate knowledge sources. Therefore we chose German to stand in as the resource-poor language L_1 and English as the resource-rich L_2; the monolingual evaluation in German will use GermaNet. The remainder of the paper describes our approach in terms of German and English, but the algorithm itself is language independent.

3.1 Concept-distance

Given a German word w^{de} in context, we use a German–English bilingual lexicon to determine its different possible English translations. Each English translation w^{en} may have one or more possible coarse senses, as listed in an English thesaurus. These English thesaurus concepts (c^{en}) will be referred to as **cross-lingual candidate senses** of the German word w^{de}.[6] Figure 1 depicts examples.[7]

As in the monolingual distributional measures, the distance between two concepts is calculated by first determining their DPs. However, in the cross-lingual approach, a concept is now glossed by near-synonymous words in an *English* thesaurus, whereas its profile is made up of the strengths of association with co-occurring *German* words. Here are constructed example cross-lingual DPs of the two senses of *star*:

Cross-lingual DPs of concepts
‘celestial body’ (*celestial body, sun, . . .*): *Raum* 0.36, *Licht* 0.27, *Konstellation* 0.11, . . .
‘celebrity’ (*celebrity, hero, . . .*): *berühmt* 0.24, *Film* 0.14, *reich* 0.14, . . .

In order to calculate the strength of association, we must first determine individual word and concept counts, as well as their co-occurrence counts.

3.2 Cross-lingual word–category co-occurrence matrix

We create a cross-lingual word–category co-occurrence matrix with German word types w^{de} as one dimension and English thesaurus concepts c^{en}

[5] The McCarthy et al. (2004) system needs to first generate a distributional thesaurus from the target text (if it is large enough—a few million words) or from another large text with a distribution of senses similar to the target text.

[6] Some of the cross-lingual candidate senses of w^{de} might not really be senses of w^{de} (e.g., ‘celebrity’, ‘river bank’, and ‘judiciary’ in Figure 1). However, as substantiated by experiments in Section 4, our algorithm is able to handle the added ambiguity.

[7] Vocabulary of German words needed to understand this discussion: ***Bank***: 1. financial institution, 2. bench (furniture); ***berühmt***: famous; ***Film***: movie (motion picture); ***Himmelskörper***: heavenly body; ***Konstellation***: constellation; ***Licht***: light; ***Morgensonne***: morning sun; ***Raum***: space; ***reich***: rich; ***Sonne***: sun; ***Star***: star (celebrity); ***Stern***: star (celestial body)

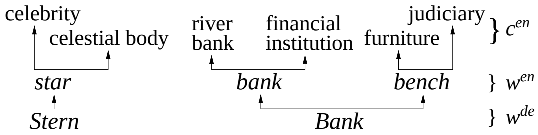


Figure 1: The cross-lingual candidate senses of German words *Stern* and *Bank*.

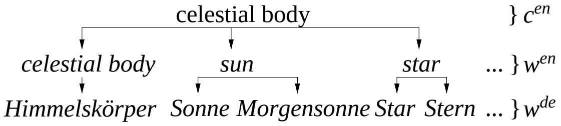


Figure 2: Words having 'celestial body' as one of their cross-lingual candidate senses.

as another.

$$\begin{array}{c|cccccc}
 & c_1^{en} & c_2^{en} & \dots & c_j^{en} & \dots \\
\hline
w_1^{de} & m_{11} & m_{12} & \dots & m_{1j} & \dots \\
w_2^{de} & m_{21} & m_{22} & \dots & m_{2j} & \dots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
w_i^{de} & m_{i1} & m_{i2} & \dots & m_{ij} & \dots \\
\vdots & \vdots & \vdots & \dots & \vdots & \ddots
\end{array}$$

The matrix is populated with co-occurrence counts from a large German corpus; we used the newspaper corpus, *taz*[8] (Sep 1986 to May 1999; 240 million words). A particular cell m_{ij}, corresponding to word w_i^{de} and concept c_j^{en}, is populated with the number of times the German word w_i^{de} co-occurs (in a window of ±5 words) with any German word having c_j^{en} as one of its *cross-lingual candidate senses*. For example, the *Raum*–'celestial body' cell will have the sum of the number of times *Raum* co-occurs with *Himmelskörper, Sonne, Morgensonne, Star, Stern*, and so on (see Figure 2). We used the *Macquarie Thesaurus* (Bernard, 1986) (about 98,000 words) for our experiments. The possible German translations of an English word were taken from the German–English bilingual lexicon BEOLINGUS[9] (about 265,000 entries).

This base word–category co-occurrence matrix (base WCCM), created after a first pass of the corpus captures strong associations between a category (concept) and co-occurring words. For example, even though we increment counts for both *Raum*–'celestial body' and *Raum*–'celebrity' for a particular instance where *Raum* co-occurs with *Star*, *Raum* will co-occur with a number of words such as *Himmelskörper, Sonne*, and *Morgensonne* that each have the sense of *celestial body* in common (see Figure 2), whereas all their other senses are likely different and distributed across the set of concepts. Therefore, the co-occurrence count of *Raum* and 'celestial body' will be relatively higher than that of *Raum* and 'celebrity'.

As in the monolingual case, a second pass of the corpus is made to disambiguate the (German) words in it. For each word, the strength of association of each of the words in its context (±5 words) with each of its cross-lingual candidate senses is summed. The sense that has the highest cumulative association with co-occurring words is chosen as the intended sense. A new bootstrapped WCCM is created by populating each cell m_{ij}, corresponding to word w_i^{de} and concept c_j^{en}, with the number of times the German word w_i^{de} co-occurs with any German word *used in cross-lingual sense* c_j^{en}. A statistic such as PMI is then applied to these counts to determine the strengths of association between a target concept and co-occurring words, giving the distributional profile of the concept.

Following the ideas described above, Mohammad et al. (2007) created Chinese–English DPCs from Chinese text, a Chinese–English bilingual lexicon, and an English thesaurus. They used these DPCs to implement an unsupervised naïve Bayes word sense classifier that placed first among all unsupervised systems taking part in the Multilingual Chinese–English Lexical Sample Task (task #5) of SemEval-07 (Jin et al., 2007).

4 Evaluation

We evaluated the newly proposed cross-lingual distributional measures of concept-distance on the tasks of (1) measuring semantic distance between German words and ranking German word pairs according to semantic distance, and (2) solving German 'Word Power' questions from *Reader's Digest*. In order to compare results with state-of-the-art monolingual approaches we conducted experiments using Ger-

[8] http://www.taz.de

[9] http://dict.tu-chemnitz.de

(Cross-lingual) Distributional Measures	**(Monolingual) GermaNet Measures**	
	Information Content–based	**Lesk-like**
α-skew divergence (Lee, 2001) (***ASD***)	Jiang and Conrath (1997) (***JC***)	hypernym pseudo-gloss (***HPG***)
cosine (Schütze and Pedersen, 1997) (***Cos***)	Lin (1998b) ($\textbf{\textit{Lin}}_{\textbf{\textit{GN}}}$)	radial pseudo-gloss (***RPG***)
Jensen-Shannon divergence (***JSD***)	Resnik (1995) (***Res***)	
Lin's measure (1998a) ($\textbf{\textit{Lin}}_{\textbf{\textit{dist}}}$)		

Table 1: Distance measures used in our experiments.

Dataset	**Year**	**Language**	**# pairs**	**PoS**	**Scores**	**# subjects**	**Correlation**
Gur65	2005	German	65	N	discrete $\{0,1,2,3,4\}$	24	.810
Gur350	2006	German	350	N, V, A	discrete $\{0,1,2,3,4\}$	8	.690

Table 2: Comparison of datasets used for evaluating semantic distance in German.

maNet measures as well. The specific distributional measures[10] and GermaNet-based measures we used are listed in Table 1. The GermaNet measures are of two kinds: (1) information content measures,[11] and (2) Lesk-like measures that rely on *n*-gram overlaps in the glosses of the target senses, proposed by Gurevych (2005)[12].

The cross-lingual measures combined the German newspaper corpus *taz* with the English *Macquarie Thesaurus* using the German–English bilingual lexicon BEOLINGUS. Multi-word expressions in the thesaurus and the bilingual lexicon were ignored. We used a context of ±5 words on either side of the target word for creating the base and bootstrapped WCCMs. No syntactic pre-processing was done, nor were the words stemmed, lemmatized, or part-of-speech tagged.

4.1 Measuring distance in word pairs

4.1.1 Data

A direct approach to evaluate distance measures is to compare them with human judgments. Gurevych (2005) and Zesch et al. (2007) asked native German speakers to mark two different sets of German word pairs with distance values. Set 1 (**Gur65**) consists of a German translation of the English Rubenstein and Goodenough (1965) dataset. It has 65 noun–noun word pairs. Set 2 (**Gur350**) is a larger dataset containing 350 word pairs made up of nouns, verbs, and adjectives. The semantically close word pairs in Gur65 are mostly synonyms or hypernyms (hyponyms) of each other, whereas those in Gur350 have both classical and non-classical relations (Morris and Hirst, 2004) with each other. Details of these **semantic distance benchmarks**[13] are summarized in Table 2. Inter-subject correlations are indicative of the degree of ease in annotating the datasets.

4.1.2 Results and Discussion

Word-pair distances determined using different distance measures are compared in two ways with the two human-created benchmarks. The rank ordering of the pairs from closest to most distant is evaluated with Spearman's rank order correlation ρ; the distance judgments themselves are evaluated with Pearson's correlation coefficient r. The higher the correlation, the more accurate the measure is. Spearman's correlation ignores actual distance values after a list is ranked—only the ranks of the two sets of word pairs are compared to determine correlation. On the other hand, Pearson's coefficient takes into account actual distance values. So even if two lists are ranked the same, but one has distances be-

[10] JSD and ASD calculate the difference in distributions of words that co-occur with the targets. Lin_{dist} (distributional measure) and Lin_{GN} (GermaNet measure) follow from Lin's (1998b) information-theoretic definition of similarity.

[11] Information content measures rely on finding the lowest common subsumer (lcs) of the target synsets in a hypernym hierarchy and using corpus counts to determine how specific or general this concept is. In general, the more specific the lcs is and the smaller the difference of its specificity with that of the target concepts, the closer the target concepts are.

[12] As GermaNet does not have glosses for synsets, Gurevych (2005) proposed a way of creating a bag-of-words-type pseudo-gloss for a synset by including the words in the synset and in synsets close to it in the network.

[13] The datasets are publicly available at: http://www.ukp.tu-darmstadt.de/data/semRelDatasets

tween consecutively-ranked word-pairs more in line with human-annotations of distance than the other, then Pearson's coefficient will capture this difference. However, this makes Pearson's coefficient sensitive to outlier data points, and so one must interpret the Pearson correlations with caution.

Table 3 shows the results.[14] Observe that on both datasets and by both measures of correlation, cross-lingual measures of concept-distance perform not just as well as the best monolingual measures, but in fact better. In general, the correlations are lower for Gur350 as it contains cross-PoS word pairs and non-classical relations, making it harder to judge even by humans (as shown by the inter-annotator correlations for the datasets in Table 2). Considering Spearman's rank correlation, α-skew divergence and Jensen-Shannon divergence perform best on both datasets. The correlations of cosine and Lin_{dist} are not far behind. Amongst the monolingual GermaNet measures, radial pseudo-gloss performs best. Considering Pearson's correlation, Lin_{dist} performs best overall and radial pseudo-gloss does best amongst the monolingual measures. Thus, we see that on both datasets and as per both measures of correlation, the cross-lingual measures perform not just as well as the best monolingual measures, but indeed slightly better.

4.2 Solving word choice problems from *Reader's Digest*

4.2.1 Data

Issues of the German edition of *Reader's Digest* include a word choice quiz called 'Word Power'. Each question has one target word and four alternative words or phrases; the objective is to pick the alternative that is most closely related to the target. The correct answer may be a near-synonym of the target or it may be related to the target by some other classical or non-classical relation (usually the former). For example:[15]

Duplikat (duplicate)

a. *Einzelstück* (single copy)	b. *Doppelkinn* (double chin)
c. *Nachbildung* (replica)	d. *Zweitschrift* (copy)

Our approach to evaluating distance measures follows that of Jarmasz and Szpakowicz (2003), who evaluated semantic similarity measures through their ability to solve synonym problems (80 TOEFL (Landauer and Dumais, 1997), 50 ESL (Turney, 2001), and 300 (English) *Reader's Digest* Word Power questions). Turney (2006) used a similar approach to evaluate the identification of semantic relations, with 374 college-level multiple-choice word analogy questions.

The **Reader's Digest Word Power (RDWP) benchmark** for German consists of 1072 of these word-choice problems collected from the January 2001 to December 2005 issues of the German-language edition (Wallace and Wallace, 2005). We discarded 44 problems that had more than one correct answer, and 20 problems that used a phrase instead of a single term as the target. The remaining 1008 problems form our evaluation dataset, which is significantly larger than any of the previous datasets employed in a similar evaluation.

We evaluate the various cross-lingual and monolingual distance measures by their ability to choose the correct answer. The distance between the target and each of the alternatives is computed by a measure, and the alternative that is closest is chosen. If two or more alternatives are equally close to the target, then the alternatives are said to be **tied**. If one of the tied alternatives is the correct answer, then the problem is counted as correctly solved, but the corresponding score is reduced. We assign a score of 0.5, 0.33, and 0.25 for 2, 3, and 4 tied alternatives, respectively (in effect approximating the score obtained by randomly guessing one of the tied alternatives). If more than one alternative has a sense in common with the target, then the thesaurus-based cross-lingual measures will mark them each as the closest sense. However, if one or more of these tied alternatives is in the same semicolon group of the thesaurus[16] as the target, then only these are chosen as the closest senses.

The German RDWP dataset contains many phrases that cannot be found in the knowledge sources (GermaNet or *Macquarie Thesaurus* via translation list). In these cases, we remove stop-

[14] In Table 3, all values are statistically significant at the 0.01 level (2-tailed), except for the one in italic (*0.212*), which is significant at the 0.05 level (2-tailed).

[15] English translations are in parentheses.

[16] Words in a thesaurus category are further partitioned into different paragraphs and each paragraph into semicolon groups. Words within a semicolon group are more closely related than those in semicolon groups of the same paragraph or category.

	Gur65		Gur350	
Measure	ρ	*r*	ρ	*r*
Monolingual				
HPG	0.672	0.702	0.346	0.331
RPG	**0.764**	0.565	**0.492**	0.420
JC	0.665	**0.748**	0.417	0.410
Lin$_{GN}$	0.607	0.739	0.475	**0.495**
Res	0.623	0.722	0.454	0.466
Cross-lingual				
ASD	**0.794**	0.597	**0.520**	0.413
Cos	0.778	0.569	0.500	*0.212*
JSD	**0.793**	0.633	**0.522**	0.422
Lin$_{dist}$	0.775	**0.816**	0.498	**0.514**

Table 3: Correlations of distance measures with human judgments.

	Reader's Digest Word Power benchmark						
Measure	**Att.**	**Cor.**	**Ties**	**Score**	**P**	**R**	**F**
Monolingual							
HPG	222	174	11	**171.5**	.77	.17	**.28**
RPG	266	188	15	**184.7**	.69	.18	**.29**
JC	357	157	1	156.0	.44	.16	.23
Lin$_{GN}$	298	153	1	152.5	.51	.15	.23
Res	299	154	33	148.3	.50	.15	.23
Cross-lingual							
ASD	438	185	81	151.6	.35	.15	.21
Cos	438	276	90	**223.1**	.51	.22	**.31**
JSD	438	276	90	**229.6**	.52	.23	**.32**
Lin$_{dist}$	438	274	90	**228.7**	.52	.23	**.32**

Table 4: Performance of distance measures on word choice problems. (Att.: Attempted, Cor.: Correct)

words (prepositions, articles, etc.) and split the phrase into component words. As German words in a phrase can be highly inflected, we lemmatize all components. For example, the target '*imaginär*' (*imaginary*) has '*nur in der Vorstellung vorhanden*' ('*exists only in the imagination*') as one of its alternatives. The phrase is split into its component words *nur, Vorstellung,* and *vorhanden*. We compute semantic distance between the target and each phrasal component and select the minimum value as the distance between target and potential answer.

4.2.2 Results and Discussion

Table 4 presents the results obtained on the German RDWP benchmark for both monolingual and cross-lingual measures. Only those questions for which the measures have some distance information are attempted; the column 'Att.' shows the number of questions attempted by each measure, which is the maximum score that the measure can hope to get. Observe that the thesaurus-based cross-lingual measures have a much larger coverage than the GermaNet-based monolingual measures. The cross-lingual measures have a much larger number of correct answers too (column 'Cor.'), but this number is bloated due to the large number of ties.[17] 'Score' is the score each measure gets after it is penalized for the ties. The cross-lingual measures *Cos*, *JSD*, and *Lin*$_{dist}$ obtain the highest scores. But 'Score' by itself does not present the complete picture either as, given the scoring scheme, a measure that attempts more questions may get a higher score just from random guessing. We therefore present precision, recall, and F-scores ($P = Score/Att$; $R = Score/1008$; $F = 2 \times P \times R/(P+R)$). Observe that the cross-lingual measures have a higher coverage (recall) than the monolingual measures but lower precision. The F scores show that the best cross-lingual measures do slightly better than the best monolingual ones, despite the large number of ties. The measures of *Cos*, *JSD*, and *Lin*$_{dist}$ remain the best cross-lingual measures, whereas HPG and RPG are the best monolingual ones.

5 Conclusion

We have proposed a new method to determine semantic distance in a possibly resource-poor language by combining its text with a knowledge source in a different, preferably resource-rich, language. Specifically, we combined German text with an English thesaurus to create cross-lingual distributional profiles of concepts—the strengths of association between English thesaurus senses (concepts) of German words and co-occurring German words—using a German–English bilingual lexicon and a bootstrapping algorithm designed to overcome ambiguities of word-senses and translations. Notably, we do so without the use of sense-annotated text or word-aligned parallel corpora. We did not parse or chunk the text, nor did we stem, lemmatize, or part-of-speech-tag the words.

We used the cross-lingual DPCs to estimate semantic distance by developing new cross-lingual

[17] We see more ties when using the cross-lingual measures because they rely on the *Macquarie Thesaurus*, a very coarse-grained sense inventory (around 800 categories), whereas the cross-lingual measures operate on the fine-grained GermaNet.

distributional measures of concept-distance. These measures are like the distributional measures of concept-distance (Mohammad and Hirst, 2006a, 2006b), except they can determine distance between words in one language using a thesaurus in a different language. We evaluated the cross-lingual measures against the best monolingual ones operating on a WordNet-like resource, GermaNet, through an extensive set of experiments on two different German semantic distance benchmarks. In the process, we compiled a large German benchmark of *Reader's Digest* word choice problems suitable for evaluating semantic-relatedness measures. Most previous semantic distance benchmarks are either much smaller or cater primarily to semantic similarity measures.

Even with the added ambiguity of translating words from one language to another, the cross-lingual measures performed better than the best monolingual measures on both the word-pair task and the *Reader's Digest* word-choice task. Further, in the word-choice task, the cross-lingual measures achieved a significantly higher coverage than the monolingual measure. The richness of English resources seems to have a major impact, even though German, with GermaNet, a well-established resource, is in a better position than most other languages. This is indeed promising, because achieving broad coverage for resource-poor languages remains an important goal as we integrate state-of-the-art approaches in natural language processing into real-life applications. These results show that our algorithm can successfully combine German text with an English thesaurus using a bilingual German–English lexicon to obtain state-of-the-art results in measuring semantic distance.

These results also support the broader and far-reaching claim that natural language problems in a resource-poor language can be solved using a knowledge source in a resource-rich language (e.g., Cucerzan and Yarowsky's (2002) cross-lingual PoS tagger). Our future work will explore other tasks such as information retrieval and text categorization. Cross-lingual DPCs also have tremendous potential in tasks inherently involving more than one language, such as machine translation and multi-language multi-document summarization. We believe that the future of natural language processing lies not in standalone monolingual systems but in those that are powered by automatically created multilingual networks of information.

Acknowledgments

We thank Philip Resnik, Michael Demko, Suzanne Stevenson, Frank Rudicz, Afsaneh Fazly, and Afra Alishahi for helpful discussions. This research is financially supported by the Natural Sciences and Engineering Research Council of Canada, the University of Toronto, the German Research Foundation under the grant "Semantic Information Retrieval" (SIR), GU 798/1-2.

References

J.R.L. Bernard, editor. 1986. *The Macquarie Thesaurus*. Macquarie Library, Sydney, Australia.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47.

Silviu Cucerzan and David Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. In *Proceedings of the 6th Conference on Computational Natural Language Learning*, pages 132–138, Taipei, Taiwan.

Christiane Fellbaum. 1998. *WordNet An Electronic Lexical Database*. MIT Press, Cambridge, MA.

James Gorman and James R. Curran. 2006. Scaling distributional similarity to large corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 361–368, Sydney, Australia.

Iryna Gurevych. 2005. Using the Structure of a Conceptual Network in Computing Semantic Relatedness. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, pages 767–778, Jeju Island, Republic of Korea.

Mario Jarmasz and Stan Szpakowicz. 2003. Roget's Thesaurus and semantic similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-2003)*, pages 212–219.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research on Computational Linguistics (ROCLING X)*, Taiwan.

Peng Jin, Yunfang Wu, and Shiwen Yu. 2007. SemEval-2007 task 05: Multilingual Chinese-English lexical sample task. In *Proceedings of the Fourth International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SemEval-07)*, Prague, Czech Republic.

Claudia Kunze, 2004. *Lexikalisch-semantische Wortnetze*, chapter Computerlinguistik und Sprachtechnologie, pages 423–431. Spektrum Akademischer Verlag.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25(2–3):259–284.

Lillian Lee. 2001. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001*, pages 65–72.

Dekang Lin. 1998a. Automatic retreival and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-98)*, pages 768–773, Montreal, Canada.

Dekang Lin. 1998b. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, San Francisco, CA. Morgan Kaufmann.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 280–267, Barcelona, Spain.

Saif Mohammad and Graeme Hirst. 2005. Distributional measures as proxies for semantic relatedness. *In submission*, http://www.cs.toronto.edu/compling/Publications.

Saif Mohammad and Graeme Hirst. 2006a. Determining word sense dominance using a thesaurus. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy.

Saif Mohammad and Graeme Hirst. 2006b. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, Sydney, Australia.

Saif Mohammad, Graeme Hirst, and Philip Resnik. 2007. Distributional profiles of concepts for unsupervised word sense disambigution. In *Proceedings of the Fourth International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SemEval-07)*, Prague, Czech Republic.

Jane Morris and Graeme Hirst. 2004. Non-classical lexical semantic relations. In *Proceedings of the Workshop on Computational Lexical Semantics, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Boston, Massachusetts.

Reinhard Rapp. 2003. Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the Machine Translation Summit IX*, pages 315–322, New Orleans, Louisiana.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 448–453, Montreal, Canada.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. *Communications of the ACM*, 8(10):627–633.

Hinrich Schütze and Jan O. Pedersen. 1997. A cooccurrence-based thesaurus and two applications to information retreival. *Information Processing and Management*, 33(3):307–318.

Peter Turney. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, Freiburg, Germany.

Peter Turney. 2006. Expressing implicit semantic relations without supervision. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 313–320, Sydney, Australia.

DeWitt Wallace and Lila Acheson Wallace. 2005. *Reader's Digest, das Beste für Deutschland*. Jan 2001–Dec 2005. Verlag Das Beste, Stuttgart.

David Yarowsky. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 454–460, Nantes, France.

Torsten Zesch, Iryna Gurevych, and Max Mühlhäuser. 2007. Comparing Wikipedia and German WordNet by evaluating semantic relatedness on multiple datasets. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2007)*, pages 205–208, Rochester, New York.

Lexical Semantic Relatedness with Random Graph Walks

Thad Hughes and Daniel Ramage
Computer Science Department
Stanford University
Stanford, CA 94305
{thughes, dramage}@cs.stanford.edu

Abstract

Many systems for tasks such as question answering, multi-document summarization, and information retrieval need robust numerical measures of lexical relatedness. Standard thesaurus-based measures of word pair similarity are based on only a single path between those words in the thesaurus graph. By contrast, we propose a new model of lexical semantic relatedness that incorporates information from every explicit or implicit path connecting the two words in the entire graph. Our model uses a random walk over nodes and edges derived from WordNet links and corpus statistics. We treat the graph as a Markov chain and compute a word-specific stationary distribution via a generalized PageRank algorithm. Semantic relatedness of a word pair is scored by a novel divergence measure, ZKL, that outperforms existing measures on certain classes of distributions. In our experiments, the resulting relatedness measure is the WordNet-based measure most highly correlated with human similarity judgments by rank ordering at $\rho = .90$.

1 Introduction

Several kinds of Natural Language Processing systems need measures of semantic relatedness for arbitrary word pairs. For example, document summarization and question answering systems often use similarity scores to evaluate candidate sentence alignments, and information retrieval systems use relatedness scores for query expansion. Several popular algorithms calculate scores from information contained in WordNet (Fellbaum, 1998), an electronic dictionary where word senses are explicitly connected by zero or more semantic relationships. The central challenge of these algorithms is to compute reasonable relatedness scores for arbitrary word pairs given that few pairs are directly connected.

Most pairs in WordNet share no direct semantic link, and for some the shortest connecting path can be surprising—even pairs that seem intuitively related, such "furnace" and "stove" share a lowest common ancestor in the hypernymy taxonomy (is-a links) all the way up at "artifact" (a man-made object). Several existing algorithms compute relatedness only by traversing the hypernymy taxonomy and find that "furnace" and "stove" are relatively unrelated. However, WordNet provides other types of semantic links in addition to hypernymy, such as meronymy (part/whole relationships), antonymy, and verb entailment, as well as implicit links defined by overlap in the text of definitional glosses. These links can provide valuable relatedness information. If we assume that relatedness is transitive across a wide variety of such links, then it is natural to follow paths such as *furnace–crematory–gas oven–oven–kitchen appliance–stove* and find a higher degree of relatedness between "furnace" and "stove."

This paper presents the application of random walk Markov chain theory to measuring lexical semantic relatedness. A graph of words and concepts is constructed from WordNet. The random walk model posits the existence of a particle that roams this graph by stochastically following local semantic relational links. The particle is biased toward exploring the neighborhood around a target word, and is allowed to roam until the proportion of time it visits each node in the limit converges to a stationary distribution. In this way we can compute distinct, word-specific probability distributions over how often a particle visits all other nodes in the graph when "starting" from a specific word. We compute the relatedness of two words as the similarity of their stationary distributions.

The random walk brings with it two distinct advantages. First, it enables the similarity measure to have a principled means of combination of multiple types of edges from WordNet. Second, by traversing all links, the walk aggregates local similarity statistics across the entire graph. The similarity scores produced by our method are, to our knowledge, the WordNet-based scores most highly correlated with human judgments.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 581–589, Prague, June 2007.

2 Related work

Budanitsky and Hirst (2006) provide a survey of many WordNet-based measures of lexical similarity based on paths in the hypernym taxonomy. As an example, one of the best performing is the measure proposed by Jiang and Conrath (1997) (similar to the one proposed by (Lin, 1991)), which finds the shortest path in the taxonomic hierarchy between two candidate words before computing similarity as a function of the information content of the two words and their lowest common subsumer in the hierarchy. We note the distinction between word similarity and word relatedness. Similarity is a special case of relatedness in that related words such as "cat" and "fur" share some semantic relationships (such as meronymy), but do not express the same likeness of form as would similar words such as "cat" and "lion." The Jiang-Conrath measure and most other measures that primarily make use of of hypernymy (is-a links) in the WordNet graph are better categorized as measures of similarity than of relatedness.

Other measures have been proposed that utilize the text in WordNet's definitional glosses, such as Extended Lesk (Banerjee and Pedersen, 2003) and later the Gloss Vectors (Patwardhan and Pedersen, 2006) method. These approaches are primarily based on comparing the "bag of words" of two synsets' gloss text concatenated with the text of neighboring words' glosses in the taxonomy. As a result, these gloss-based methods measure relatedness. Our model captures some of this relatedness information by including weighted links based on gloss text.

A variety of other measures of semantic relatedness have been proposed, including distributional similarity measures based on co-occurrence in a body of text—see (Weeds and Weir, 2005) for a survey. Other measures make use of alternative structured information resources than WordNet, such as Roget's thesaurus (Jarmasz and Szpakowicz, 2003). More recently, measures incorporating information from Wikipedia (Gabrilovich and Markovitch, 2007; Strube and Ponzetto, 2006) have reported stronger results on some tasks than have been achieved by existing measures based on shallower lexical resources. The results of our algorithm are competitive with some Wikipedia algorithms while using only WordNet 2.1 as the underlying lexical resource. The approach presented here is generalizable to construction from any underlying semantic resource.

PageRank is the most well-known example of a random walk Markov chain—see (Berkhin, 2005) for a survey. It uses the local hyperlink structure of the web to define a graph which it walks to aggregate popularity information for different pages. Recent work has applied random walks to NLP tasks such as PP attachment (Toutanova et al., 2004), word sense disambiguation (Mihalcea, 2005; Tarau et al., 2005), and query expansion (Collins-Thompson and Callan, 2005). However, to our knowledge, the literature in NLP has only considered using one stationary distribution per specially-constructed graph as a probability estimator. In this paper, we introduce a measure of semantic relatedness based on the divergence of the distinct stationary distributions resulting from random walks centered at different positions in the word graph. We believe we are the first to define such a measure.

3 Random walks on WordNet

Our model is based on a random walk of a particle through a simple directed graph $G = (V, E)$ whose nodes V and edges E are extracted from WordNet version 2.1. Formally, we define the probability $n_i^{(t)}$ of finding the particle at node $n_i \in V$ at time t as the sum of all ways in which the particle could have reached n_i from any other node at the previous time-step:

$$n_i^{(t)} = \sum_{n_j \in V} n_j^{(t-1)} P(n_i \mid n_j)$$

where $P(n_i \mid n_j)$ is the conditional probability of moving to n_i given that the particle is at n_j. In particular, we construct the transition distribution such that $P(n_i \mid n_j) > 0$ whenever WordNet specifies a local link relationship of the form $j \rightarrow i$. Note that this random walk is a Markov chain because the transition probabilities at time t are independent of the particle's past trajectory.

The subsections that follow present the construction of the graph for our random walk from WordNet and the mathematics of computing the stationary distribution for a given word.

3.1 Graph Construction

WordNet is itself a graph over synsets. A synset is best thought of as a concept evoked by one sense of one or more words. For instance, different senses of the word "bank" take part in different synsets (e.g. a river bank versus a financial institution), and a single synset can be represented by multiple synonymous words, such as "middle" and "center." WordNet explicitly marks semantic relationships between synsets, but we are additionally interested in representing relatedness between words. We therefore extract the following types of nodes from WordNet:

Synset Each WordNet synset has a corresponding node. For example, one node corresponds to the synset referred to by "dog#n#3," the third sense of dog as noun, whose meaning is "an informal term for a man." There are 117,597 *Synset* nodes.

TokenPOS One node is allocated to every word coupled with a part of speech, such as "dog#n" meaning dog as a noun. These nodes link to all the synsets they participate in, so that "dog#n" links the *Synset* nodes for canine, hound, hot dog, etc. Collocations—multi-word expressions such as "hot dog"—that take part in a synsets are also represented by these nodes. There are 156,588 *TokenPOS* nodes.

Token Every *TokenPOS* is connected to a *Token* node corresponding to the word when no part of speech information is present. For example, "dog" links to "dog#n" and "dog#v" (meaning "to chase"). There are 148,646 *Token* nodes.

Synset nodes are connected with edges corresponding to many of the relationship types in WordNet. We use these WordNet relationships to form edges: hypernym/hyponym, instance/instance of, all holonym/meronym links, antonym, entails/entailed by, adjective satellite, causes/caused by, participle, pertains to, derives/derived from, attribute/has attribute, and topical (but not regional or usage) domain links. By construction, each edge created from a WordNet relationship is guaranteed to have a corresponding edge in the opposite direction.

Edges that connect a *TokenPOS* to the *Synsets* using it are weighted based on a Bayesian estimate drawn from the SemCor frequency counts included in WordNet but with a non-uniform Dirichlet prior. Our edge weights are the SemCor frequency counts for each target *Synset,* with pseudo-counts of .1 for all *Synsets*, 1 for first sense of each word, and .1 for the first word in each *Synset*. Intuitively, this causes the particle to have a higher probability of moving to more common senses of a *TokenPOS*; for example, the edges from "dog#n" to "dog#n#1" (canine) and "dog#n#5" (hotdog) have un-normalized weights of 43.2 and 0.1, respectively. The edges connecting a *Token* to the *TokenPOS* nodes in which it can occur are also weighted by the sum of the weights of the outgoing *TokenPOS→Synset* links. Hence a walk starting at a common word like "cat" is far more likely to follow a link to "cat#n" than to rarities like "cat#v" (to vomit). These edges are uni-directional; no edges are created from a *Synset* to a *TokenPOS* that can represent the *Synset*.

In order for our graph construction to incorporate textual gloss-based information, we also create uni-directional edges from *Synset* nodes to the *TokenPOS* nodes for the words and collocations used in that synset's gloss definition. This requires part-of-speech tagging the glosses, for which we use the Stanford maximum entropy tagger (Toutanova et al., 2003). It is important to correctly weight these edges, because high-frequency stop-words such as "by" and "he" do not convey much information and might serve only to smear the probability mass across the whole graph. Gloss-based links to these nodes should therefore be down-weighted or removed. On the other hand, up-weighting extremely rare words such as by `tf-idf` scoring might also be inappropriate because such rare words would get extremely high scores, which is an undesirable trait in similarity search. (Haveliwala et al., 2002) and others have shown that a "non-monotonic document frequency" (NMDF) weighting can be more effective in such a setting. Because the frequency of words in the glosses is distributed by a power-law, we weight each word by its distance from the mean word count in log space. Formally, the weight w_i for a word appearing r_i times is

$$w_i = exp\left(-\frac{(log(r_i) - \mu)^2}{2\sigma^2}\right)$$

where μ and σ are the mean and standard deviation of the logs of all word counts. This is a smooth approximation to the high and low frequency stop lists used effectively by other measures such as (Patwardhan and Pedersen, 2006). We believe that because non-monotonic frequency scaling has no parameters and is data-driven, it could stand to be more widely adopted among gloss-based lexical similarity measures.

We also add bi-directional edges between *Synsets* whose word senses overlap with a common *TokenPOS*. These edges have raw weights given by the number of *TokenPOS* nodes shared by the *Synsets*. The intuition behind adding these edges is that WordNet often divides the meanings of words into fine-grained senses with similar meanings, so there is likely to be some semantic relationship between *Synsets* sharing a common *TokenPOS*.

The final graph has 422,831 nodes and 5,133,281 edges. This graph is very sparse; fewer than 1 in 10,000 node pairs are directly connected. When only the unweighted WordNet relationship edges are considered, the largest degree of any node is "city#n#1" with 667 edges (mostly connecting to particular cities), followed by "law#n#2" with 602 edges (mostly connecting to a large number of domain terms such as "dissenting opinion" and "freedom of speech"), and each node is on average connected to 1.7 other nodes. When the gloss-based edges are considered separately, the highest degree nodes are those with the longest definitions; the maximum out-degree is 56 and the average out-degree is 6.2. For the edges linking *TokenPOS* nodes to the *Synsets* in which they participate, *TokenPOS* nodes with many senses are the most connected; "break#v" with 59 outgoing edges and "make#v" with 49 outgoing edges have the highest out-degrees, with the average out-degree being 1.3.

3.2 Computing the stationary distribution

Each of the K edge types presented above can be represented as separate transition matrix $E_k \in \mathbb{R}^{N \times N}$ where

N is the total number of nodes. For each matrix, column j contains contains a normalized outgoing probability distribution,[1] so the weight in cell (i, j) contains $P_K(n_i \mid n_j)$, the conditional probability of moving from node n_j to node n_i in edge type K. For many of the edge types, this is either 0 or 1, but for the weighted edges, these are real valued. The full transition matrix M is then the column normalized sum of all of the edge types:

$$\hat{M} = \sum_k E_k$$
$$M = \left(\left\|\hat{M}\right\|_\infty\right)^{-1} \cdot \hat{M}$$

$M$is a distillation of relevant relatedness information about all nodes extracted from WordNet and is not tailored for computing a stationary distribution for any specific word. In order to compute the stationary distribution $v_{dog\#n}$ for a walk centered around the *TokenPOS* "dog#n," we first define an initial distribution $v^{(0)}_{dog\#n}$ that places all the probability mass in the single vector entry corresponding to "dog#n." Then at every step of the walk, we will return to $v^{(0)}$ with probability β. Intuitively, this return probability captures the notion that nodes close to "dog#n" should be given higher weight, and also guarantees that the stationary distribution exists and is unique (Bremaud, 1999). The stationary distribution v is computed via an iterative update algorithm:

$$v^{(t)} = \beta v^{(0)} + (1 - \beta) M v^{(t-1)}$$

Because the walk may return to the initial distribution $v^{(0)}$ at any step with probability β, we found that $v^{(t)}$ converges to its unique stationary distribution $v^{(\infty)}$ in a number of steps roughly proportional to β^{-1}. We experimented with a range of return probabilities and found that our results were relatively insensitive to this parameter. Our convergence criteria was $\left\|v^{(t-1)} - v^{(t)}\right\|_1 < 10^{-10}$, which, for our graph with a return probability of $\beta = .1$, was met after about two dozen iterations. This computation takes under two seconds on a modern desktop machine.

Note that because M is sparse, each iteration of the above computation is linear in the total number of nonzero entries in P, i.e. linear in the total number of edges. Introducing an edge type that is dense would dramatically increase running time.

3.3 Model variants

For this paper, we consider three model variants that differ based on which subset of the edge types are included in the transition matrix M.

MarkovLink This variant includes the explicit WordNet relations such as hypernymy and the edges representing overlap between the *TokenPOS* nodes contained in *Synsets*. A particle walking through this graph reaches only *Synset* nodes and can step from one *Synset* to another whenever WordNet specifies a relationship between the *Synsets* or when the *Synsets* share a common word. There is a single connected component in this model variant. This model is loosely analogous to a smoothed version of the path-based WordNet measures surveyed in (Budanitsky and Hirst, 2006) but differs in that it integrates multiple link types and aggregates relatedness information across all paths in the graph.

MarkovGloss This variant includes only the weighted uni-directional edges linking *Synsets* to the *TokenPOS* nodes contained in their gloss definitions, and the edges from a *TokenPOS* node to the *Synsets* containing it. The intuition behind this model variant is that the particle can move as if it were recursively looking up words in a dictionary, stepping from *Synsets* to the *Synsets* used to define them. Because WordNet's gloss definitions are not sense-tagged, the particle must make an intermediate step to a *TokenPOS* contained in the gloss definition and then to a *Synset* representing a particular sense of that *TokenPOS*. The availability of sense-tagged glosses would eliminate the noise introduced by this intermediate step. The particle can reach both *Synsets* and *TokenPOS* nodes in this variant, but some parts of the graph are not reachable from other parts. This model incorporates much of the same information as the gloss-based WordNet measures (Banerjee and Pedersen, 2003; Patwardhan and Pedersen, 2006) but differs in that it considers many more glosses than just those in the immediate neighborhoods of the candidate words.

MarkovJoined This variant is the natural combination of the above two; we construct the graph containing WordNet relation edges, *Synset* overlap edges, and gloss-based *Synset* to *TokenPOS* edges.

Many of the characteristics of the model variants can be understood in terms of how much probability mass they assign to each node for a particular word-specific stationary distribution. Table 1 shows the highest scoring nodes in the word-specific stationary distributions centered around the *Token* node for "wizard," as computed by the *MarkovLink* and *MarkovGloss* variants. In both variants, the "wizard" *Token*'s only neighbors are the "wizard#n" and "wizard#a" *TokenPOS* nodes, and "wizard#n"

[1] The frequency-count derived edges are normalized by the largest column sum. This effectively preserves relative term frequency information across the graph and causes some columns to sum to less than one. We interpret this lost mass as a link to "nowhere."

MarkovLink		*MarkovGloss*	
Node	Probability	Node	Probability
wizard	1.0E-1	wizard	1.3E-01
wizard#n	2.5E-3	wizard#n	2.9E-02
wizard#a	7.8E-5	wizard#a	9.1E-04
ace#n#3	4.2E-5	ace#n#3	1.1E-06
sorcerer#n#1	2.2E-6	sorcerer#n#1	5.8E-07
charming#a#2	2.2E-6	dazzlingly#r	2.4E-08
expert#n#1	1.1E-6	charming#a#2	1.6E-09
track star#n#1	1.1E-6	sorcery#n	2.6E-10
occultist#n#1	5.7E-7	magic#n	6.8E-12
Cagliostro#n#1	5.7E-7	magic#a	6.8E-12
star#v#2	5.5E-7	dazzlingly#r#1	4.3E-14
breeze_through#v#1	5.4E-7	dazzle#n	9.4E-16
magic#n#1	2.1E-8	beholder#n	9.4E-16
sorcery#n#1	2.1E-7	dazzle#v	9.4E-16
magician#n#1	1.9E-7	magic#n#1	5.1E-16

Table 1: Highest scoring nodes in the stationary distributions for "wizard#n" as generated by the *MarkovLink* model and the *MarkovGloss* model with return probability 0.1.

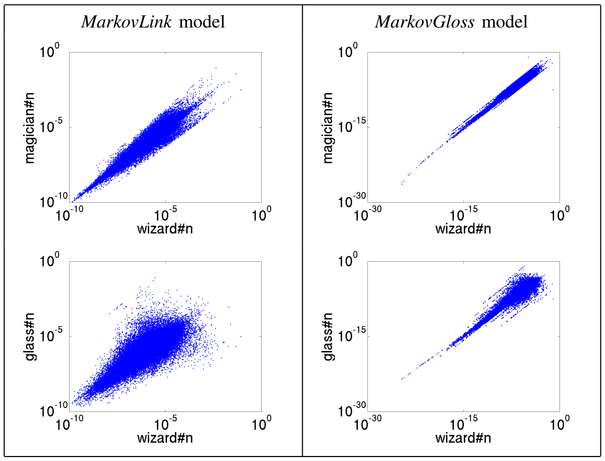


Figure 1: Example stationary distributions plotted against each other for similar (top) and dissimilar (bottom) word pairs, using the *MarkovLink* (left) and *MarkovGloss* (right) model variants.

has a higher probability mass because of its higher SemCor usage counts. Likewise, the only possible steps permitted in either variant from "wizard#n" and "wizard#a" are to the *Synsets* that can be expressed with those nodes: "ace#n#3," "sorcerer#n#1," and "charming#a#1." Again, the amount of mass given to these nodes depends on the strength of these edge weights, which is determined by the SemCor usage counts.

The highest probability nodes in the table are common because both model variants share the same initial links. However, the orders of the remaining nodes in the stationary distributions are different. In the *MarkovLink* variant, the random walk can only proceed to other *Synsets* using WordNet relationship edges; "track star#n#1" and "expert#n#1" are first reached by following hyponym and hypernym edges from "ace#n#1," and "occultist#n#1" and "Cagliostro#n#1" are first reached with hypernym and instance edges from "sorcerer#n#1." The node "breeze through#v#1" is reached through a path following derivational links with "ace#n" and "ace#v."

The *MarkovGloss* variant in table 1 shows how information can be extracted solely from the textual glosses. Once the random walk reaches the first *Synset* nodes, it can step to the *TokenPOS* nodes in their glosses; for example, "ace#n#1" has the gloss "someone who is dazzlingly skilled in any field." Links to *TokenPOS* nodes that are very common in glosses are down-weighted with NMDF weighting, so "someone#n" receives little mass while "dazzlingly#r" receives more. From there, the random walk can step to another *Synset* such as "dazzlingly#r#1," and then on to other *TokenPOS* nodes used in its definition: "in a manner or to a degree that dazzles the beholder."

Figure 1 demonstrates how two word-specific stationary distributions are more highly correlated if the words are related. In both model variants, random walks for related words are more likely to visit the same parts of the graph, and so assign higher probability to the same nodes. Figure 1 also shows that the *MarkovGloss* variant produces distributions with a much wider range of probabilities than the *MarkovLink*, which might be a source of difficulty in integrating the two model variants.

Figure 2 shows the correlation between the stationary distributions produced by the two model variants for the same word. The log-log scale makes it possible to see the entire range of probabilities on the same axes, and shows that distributions produced by these two model variants share many of the same highest-probability words.

A noteworthy property of the constructed graphs is that word relatedness can be computed directly by comparing walks that start at *Token* nodes. By contrast, existing WordNet-based measures require independent similarity judgments for all word senses relevant to a target word pair (of which the maximum relatedness value is usually taken). Our algorithm lends itself to comparisons between walks centered at a *Synset* node, or a *TokenPOS* node, or a *Token* node, or any mixed distribution thereof. And because the *Synset* nodes are strongly connected, the model also admits direct comparison across parts of speech.

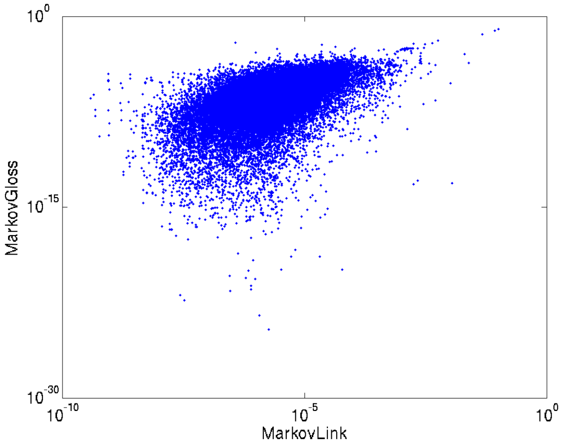


Figure 2: Correlation of the stationary distributions for "wizard#n," produced by the *MarkovLink* variant (x-axis) and the *MarkovGloss* variant (y-axis).

4 Similarity judgments

We have shown how to compute the word-specific stationary distribution from any starting distribution in the graph. Now consider the task of deciding similarity between two words. Intuitively, if the random walk starting at the first word's node and the random walk starting at the second word's node tend to visit the same nodes, we would like to consider them semantically related. Formally, we measure the divergence of their respective stationary distributions, p and q.

A wide literature exists on similarity measures between probability distributions. One standard choice is to consider p and q to be vectors and measure the cosine of the angle between them, which is rank equivalent to Euclidean distance.

$$sim_{cos}(p, q) = \frac{\sum_i p_i q_i}{\|p\| \; \|q\|}$$

Because p and q are probability distributions, we would also expect a strong contender from the information-theoretic measures based on Kullback-Leibler divergence, defined as:

$$D_{KL}(p \parallel q) = \sum_i p_i \, log \frac{p_i}{q_i}$$

Unfortunately, KL divergence is undefined if any q_i is zero because those terms in the sum will have infinite weight. Several modifications to avoid this issue have been proposed in the literature. One is Jensen-Shannon divergence (Lin, 1991), a symmetric measure based on KL-divergence defined as the average of the KL divergences of each distribution to their average distribution. Jensen-Shannon is well defined for all distributions because the average of p_i and q_i is non-zero whenever either number is.

These measures and others are surveyed in (Lee, 2001), who finds that Jensen-Shannon is outperformed by the Skew divergence measure introduced by Lee in (1999). The skew divergence[2] accounts for zeros in q by mixing in a small amount of p.

$$\begin{aligned} s_\alpha(p, q) &= D(p \parallel \alpha q + (1-\alpha)p) \\ &= \textstyle\sum_i p_i \, log \frac{p_i}{\alpha q_i + (1-\alpha) p_i} \end{aligned}$$

Lee found that as $\alpha \rightarrow 1$, the performance of skew divergence on natural language tasks improves. In particular, it outperforms most other models and even beats pure KL divergence modified to avoid zeros with sophisticated smoothing models. In exploring the performance of divergence measures on our model's stationary distributions, we observed the same phenomenon. Note that in the limit as $\alpha \rightarrow 1$, alpha skew is identically KL-divergence.

4.1 Zero-KL Divergence

In this section we introduce a novel measure of distributional divergence based on a reinterpretation of the skew divergence. Skew divergence avoids zeros in q by mixing in some of p, but its performance on many natural language tasks improves as it better approximates KL divergence. We propose an alternative approximation to KL divergence called Zero-KL divergence, or ZKL. When q_i is non-zero, we use exactly the term from KL divergence. When $q_i = 0$, we have a problem—in the limit as $\alpha \rightarrow 1$, the corresponding term approaches infinity. We let ZKL use the Skew divergence value for these terms: $p_i \, log \frac{p_i}{\alpha q_i + (1-\alpha) p_i}$. Because $q_i = 0$ this simplifies to $p_i \, log \frac{p_i}{(1-\alpha) p_i} = p_i \, log \frac{1}{1-\alpha}$.

Lee showed skew divergence's best performance was for α near to 1, so we formalize this intuition by choosing α exponentially near to 1, i.e. we can choose our α as $1 - 2^{-\gamma}$ for some $\gamma \in \mathbb{R}^+$. Zero terms in the sum can now be written as $p_i \, log \frac{1}{2^{-\gamma}} = p_i \, log \, 2^\gamma = p_i \, \gamma$. Note here an analogy to the case with $q_j > 0$ and where p_j is exactly one order of magnitude greater than q_j, i.e. $p_j = 2 \cdot q_j$. For such a term in the standard KL divergence, we would get $p_j \, log \frac{p_j}{q_j} = p_j \, log(2) = p_j$. Therefore, the α term in skew divergence implicitly defines a parameter stating how many orders of magnitude smaller than p_j to count q_j if $q_j = 0$.

We define the Zero-KL divergence with respect to

[2] In Lee's (1999) original presentation, skew divergence is defined not as $s_\alpha(p, q)$ but rather as $s_\alpha(q, p)$. We reverse the argument order for consistency with the other measures discussed here.

gamma:

$$ZKL_\gamma(p,q) = \sum_i p_i \left\{ \begin{array}{ll} log\frac{p_i}{q_i} & q_i \neq 0 \\ \gamma & q_i = 0 \end{array} \right.$$

Note that this is exactly KL-divergence when KL-divergence is defined and, like skew divergence, approximates KL divergence in the limit as $\gamma \rightarrow \infty$.

A similar analysis of the skew divergence terms for when $0 < q_i \ll p_i$ (and in particular with q_i less than p_i by more than a factor of $2^{-\gamma}$) shows that such a term in the skew divergence sum is again approximated by γp_i. ZKL does not have this property. Because ZKL is a better approximation to KL divergence and because they have the same behavior in the limit, we expect ZKL's performance to dominate that of skew divergence in many distributions. However, if there is a wide range in the exponent of noisy terms, the maximum possible penalty to such terms ascribed by skew divergence may be beneficial.

Figure 3 shows the relative performance of ZKL versus Jensen-Shannon, skew divergence, cosine similarity, and the Jaccard score (a measure from information retrieval) for correlations with human judgment on the *MarkovLink* model. ZKL consistently outperforms the other measures on distributions resulting from this model, but ZKL is not optimal on distributions generated by our other models. The next section explores this topic in more detail.

5 Evaluation

Traditionally, there have been two primary types of evaluation for measures of semantic relatedness: one is correlation to human judgment, the other is the relative performance gains of a task-driven system when it uses the measure. The evaluation here focuses on correlation with human judgments of relatedness. For consistency with previous literature, we use rank correlation (Spearman's ρ coefficient) rather than linear correlation when comparing sets of relatedness judgments because the rank correlation captures information about the relative ordering of the scores. However, it is worth noting that many applications that make use of lexical relatedness scores (e.g. as features to a machine learning algorithm) would better be served by scores on a linear scale with human judgments.

Rubenstein and Goodenough (1965) solicited human judgments of semantic similarity for 65 pairs of common nouns on a scale of zero to four. Miller and Charles (1991) repeated their experiment on a subset of 29 noun pairs (out of 30 total) and found that although individuals varied among their judgments, in aggregate the scores were highly correlated with those found by Rubenstein and Goodenough (at $\rho = .944$ by our calculation). Resnik (1999) replicated the Miller and Charles experiment and reported that the average per-subject linear correlation on the dataset was around $r = 0.90$, providing a rough upper bound on any system's linear correlation performance with respect to the Miller and Charles data. Figure 3 shows that the ZKL measure on the *MarkovLink* model has linear correlation coefficient $r = .903$—at the limit of human inter-annotator agreement.

Recently, a larger set of word relatedness judgments was obtained by (Finkelstein et al., 2002) in the WordSimilarity-353 (WS-353) collection. Despite the collection's name, the study instructed participants to score word pairs for relatedness (on a scale of 0 to 10), which is in contrast to the similarity judgments requested of the Miller and Charles (MC) and Rubenstein and Goodenough (RG) participants. For this reason, the WordSimilarity-353 data contains many pairs that are not semantically similar but still receive high scores, such as "computer-software" at 8.81. WS-353 contains pairs that include non-nouns, such as "eat-drink," one proper noun not appearing in WordNet ("Maradona-football"), and some pairs potentially subject to political bias. Again, the aggregate human judgments correlate well with earlier data sets where they overlap—the 30 judgments that WordSimilarity-353 shares with the Miller and Charles data have $\rho = .939$ and the 29 shared with Rubenstein and Goodenough have $\rho = .904$ (by our calculations).

We generated similarity scores for word pairs in all three data sets using the three variants of our walk model (*MarkovLink, MarkovGloss, MarkovJoined*) and with multiple distributional distance measures. We used the WordNet::Similarity package (Pedersen et al., 2004) to compute baseline scores for several existing measures, noting that one word pair was not processed in WS-353 because one of the words was missing from WordNet. The results are summarized in Table 2. These numbers differ slightly from previously reported scores due to variations in the exact experimental setup, WordNet version, and the method of breaking ties when computing ρ (here we break ties using the product-moment formulation of Spearman's rank correlation coefficient). It is worth noting that in their experiments, (Patwardhan and Pedersen, 2006) report that the Vector method has rank correlation coefficients of .91 and .90 for MC and RG, respectively, which are also top performing values.

In our experiments, the *MarkovLink* model with ZKL distance measure was the best performing model overall. *MarkovGloss* and *MarkovJoined* were also strong contenders but with the cosine measure instead of ZKL. One reason for this distinction is that the stationary distributions resulting from the *MarkovLink* model are non-zero for all but the initial word nodes (i.e. non-zero for all *Synset* nodes). Consequently, ZKL's re-estimate for the zero terms adds little information. By contrast, the *MarkovGloss* and *MarkovJoined* models include links that traverse from *Synset* nodes to *TokenPOS* nodes, re-

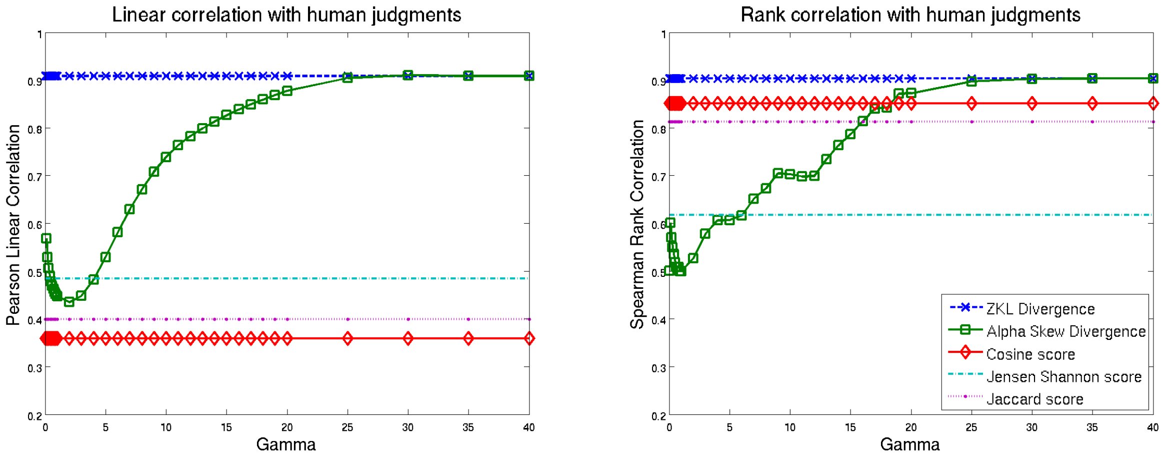


Figure 3: Correlation with the Miller & Charles data sets by linear correlation (left) and rank correlation (right) for the *MarkovLink* model. All data points were based on one set of stationary distributions over the graph; only the divergence measure between those distributions is varied. Note that ZKL_γ dominates both graphs but skew divergence does well for increasing α (computed as $1 - 2^\gamma$). Gamma is swept over the range 0 to 1, then 1 through 20, then 20 through 40 at equal resolutions.

Model	MC Rank	RG Rank	WS-353 Rank
MarkovLink (ZKL)	**.904**	.817	**.552**
MarkovGloss (cosine)	.841	.762	.467
MarkovJoined (cosine)	.841	**.838**	.547
Gloss Vectors	.888	.789	.445
Extended Lesk	.869	.829	.511
Jiang-Conrath	.653	.584	.195
Lin	.625	.599	.216

Table 2: Spearman's ρ rank correlation coefficients with human judgments using $\gamma = 2.0$ for ZKL. Note that figure 3 demonstrates ZKL's insensitivity with regard to the parameter setting for the *MarkovLink* model.

sulting in a final stationary distribution with more (and more meaningful) zero/non-zero pairs. Hence the proper setting of gamma (or alpha for skew divergence) is of greater importance. ZKL's performance improves with tuning of gamma, but cosine similarity remained the more robust measure for these distributions.

6 Conclusion

In this paper, we have introduced a new measure of lexical relatedness based on the divergence of the stationary distributions computed from random walks over graphs extracted WordNet. We have explored the structural properties of extracted semantic graphs and characterized the distinctly different types of stationary distributions that result. We explored several distance measures on these distributions, including ZKL, a novel variant of KL-divergence. Our best relatedness measure is at the limit of human inter-annotator agreement and is one of the strongest measures of semantic relatedness that uses only WordNet as its underlying lexical resource.

In future work, we hope to integrate other lexical resources such as Wikipedia into the walk. Incorporating more types of links from more resources will underline the importance of determining appropriate relative weights for all of the types of edges in the walk's matrix. Even for WordNet, we believe that certain link types, such as antonyms, may be more or less appropriate for certain tasks and should weighted accordingly. And while our measure of lexical relatedness correlates well with human judgments, we hope to show performance gains in a real-word task from the use of our measure.

Acknowledgments

Thanks to Christopher D. Manning and Dan Jurafsky for their helpful comments and suggestions. We are also grateful to Siddharth Patwardhan and Ted Pedersen for assistance in comparing against their system. Thanks to Sushant Prakash, Rion Snow, and Varun Ganapathi for their advice on pursuing some of the ideas in this paper, and to our anonymous reviewers for their helpful critiques. Daniel Ramage was funded in part by an NDSEG fellowship. This work was also supported in part by the DTO AQUAINT Program, the DARPA GALE Program, and the ONR (MURI award N000140510388).

References

S. Banerjee and T. Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco*, pages 805–810.

P. Berkhin. 2005. A survey on pagerank computing. *Internet Mathematics*, 2(1):73–120.

P. Bremaud. 1999. *Markov chains: Gibbs fields, monte carlo simulation, and queues.* Springer-Verlag.

A. Budanitsky and G. Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

K. Collins-Thompson and J. Callan. 2005. Query expansion using random walk models. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 704–711, New York, NY, USA. ACM Press.

C. Fellbaum. 1998. *WordNet: An electronic lexical database.* MIT Press.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*.

T. Haveliwala, A. Gionis, D. Klein, and P. Indyk. 2002. Evaluating strategies for similarity search on the web. In *WWW2002*.

Mario Jarmasz and Stan Szpakowicz. 2003. Roget's thesaurus and semantic similarity. In *Proceedings of RANLP-03*, pages 212–219.

J. J. Jiang and D. W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X)*, pages 19–33.

Lillian Lee. 1999. Measures of distributional similarity. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

Lillian Lee. 2001. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001*, pages 65–72.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. In *IEEE Transactions on Information Theory*, volume 37(1), pages 145–151.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 411–418, Morristown, NJ, USA. Association for Computational Linguistics.

G.A. Miller and W.G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6:1–28.

S. Patwardhan and T. Pedersen. 2006. Using wordnet-based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of the EACL 2006 Workshop Making Sense of Sense - Bringing Computational Linguistics and Pyscholinguistics Together*, pages 1–8.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*.

Philip Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, (11):95–130.

H. Rubenstein and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Computational Linguistics*, 8:627–633.

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1419–1424.

Paul Tarau, Rada Mihalcea, and Elizabeth Figa. 2005. Semantic document engineering with wordnet and pagerank. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 782–786, New York, NY, USA. ACM Press.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, New York, NY, USA. ACM Press.

Julie Weeds and David Weir. 2005. Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Comput. Linguist.*, 31(4):439–475.

Experimental Evaluation of LTAG-based Features for Semantic Role Labeling

Yudong Liu and **Anoop Sarkar**
Simon Fraser University
Burnaby, BC, Canada
{yudongl,anoop}@cs.sfu.ca

Abstract

This paper proposes the use of Lexicalized Tree-Adjoining Grammar (LTAG) formalism as an important additional source of features for the Semantic Role Labeling (SRL) task. Using a set of one-vs-all Support Vector Machines (SVMs), we evaluate these LTAG-based features. Our experiments show that LTAG-based features can improve SRL accuracy significantly. When compared with the best known set of features that are used in state of the art SRL systems we obtain an improvement in F-score from 82.34% to 85.25%.

1 Introduction

Semantic Role Labeling (SRL) aims to identify and label all the arguments for each predicate occurring in a sentence. It involves identifying constituents in the sentence that represent the predicate's arguments and assigning pre-specified semantic roles to them.

[A0$_{seller}$ *Ports of Call Inc.*] reached agreements to [V$_{\text{verb}}$ *sell*] [A1$_{\text{thing}}$ *its remaining seven aircraft*] [A2$_{\text{buyer}}$ *to buyers that weren't disclosed*] .

is an example of SRL annotation from the PropBank corpus (Palmer et al., 2005), where the subscripted information maps the semantic roles A0, A1, A2 to arguments for the predicate *sell* as defined in the PropBank Frame Scheme. For SRL, high accuracy has been achieved by:
(i) proposing new types of features (see Table 1 in Section 3 for previously proposed features),
(ii) modeling the predicate frameset by capturing dependencies between arguments (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Toutanova et al., 2005; Punyakanok et al., 2005a),
(iii) dealing with incorrect parser output by using more than one parser (Pradhan et al., 2005b).

Our work in this paper falls into category (i). We propose several novel features based on Lexicalized Tree Adjoining Grammar (LTAG) derivation trees in order to improve SRL performance. To show the usefulness of these features, we provide an experimental study comparing LTAG-based features with the standard set of features and kernel methods used in state-of-the-art SRL systems. The LTAG formalism provides an extended domain of locality in which to specify predicate-argument relationships and also provides the notion of a derivation tree. These two properties of LTAG make it well suited to address the SRL task.

SRL feature extraction has relied on various syntactic representations of input sentences, such as syntactic chunks (Hacioglu et al., 2004) and full syntactic parses (Gildea and Jurafsky, 2002). In contrast with features from shallow parsing, previous work (Gildea and Palmer, 2002; Punyakanok et al., 2005b) has shown the necessity of full syntactic parsing for SRL. In order to generalize the *path* feature (see Table 1 in Section 3) which is probably the most salient (while being the most data sparse) feature for SRL, previous work has extracted features from other syntactic representations, such as CCG derivations (Gildea and Hockenmaier, 2003) and dependency trees (Hacioglu, 2004) or integrated features from different parsers (Pradhan et al., 2005b). To avoid explicit feature engineering on trees, (Moschitti, 2004) used convolution kernels on selective portions of syntactic trees. In this paper, we also compare our work with tree kernel based methods.

Most SRL systems exploit syntactic trees as the main source of features. We would like to take this one step further and show that using LTAG deriva-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 590–599, Prague, June 2007.

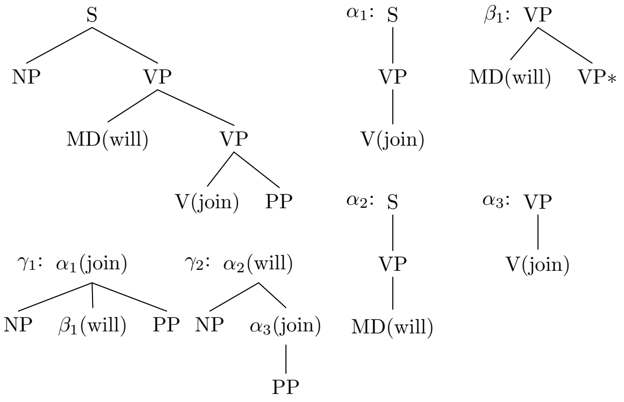


Figure 1: A parse tree schematic, and two plausible LTAG derivation trees for it: derivation tree γ_1 uses elementary trees α_1 and β_1 while γ_2 uses α_2 and α_3.

tion trees as an additional source of features can improve both argument identification and classification accuracy in SRL.

2 Using LTAG-based Features in SRL

We assume some familiarity with Lexicalized Tree-Adjoining Grammar (LTAG); (Joshi and Schabes, 1997) is a good introduction to this formalism. A LTAG is defined to be a set of lexicalized *elementary trees* (etree for short), of which there are two types, *initial trees* and *auxiliary trees*. Typically etrees can be composed through two operations into parse trees, *substitution* and *adjunction*. We use *sister adjunction* which is commonly used in LTAG statistical parsers to deal with the relatively flat Penn Treebank trees (Chiang, 2000). The tree produced by composing the etrees is the *derived/parse tree* and the tree that records the history of composition is the *derivation tree*.

A reasonable way to define SRL features is to provide a strictly local dependency (i.e. within a single etree) between predicate and argument. There have been many different proposals on how to maintain syntactic locality (Xia, 1999; Chen and Vijay-Shanker, 2000) and SRL locality (Chen and Rambow, 2003; Shen and Joshi, 2005) when extracting LTAG etrees from a Treebank. These proposed methods are exemplified by the derivation tree γ_1 in Fig. 1. However, in most cases they can only provide a local dependency between predicate and argument for 87% of the argument constituents (Chen and Rambow, 2003), which is too low to provide high SRL accuracy. In LTAG-based statistical parsers, high accuracy is obtained by using the Magerman-Collins head-percolation rules in order to provide the etrees (Chiang, 2000). This method is exemplified by the derivation tree γ_2 in Fig. 1. Comparing γ_1 with γ_2 in Fig. 1 and assuming that *join* is the predicate and the *NP* is the potential argument, the path feature as defined over the LTAG *derivation tree* γ_2 is more useful for the SRL task as it distinguishes between main clause and non-finite embedded clause predicates. This alternative derivation tree also exploits the so-called *extended domain of locality* (Joshi and Schabes, 1997) (the examples in Section 2.1 show this clearly). In this paper, we crucially rely on features defined on LTAG derivation trees of the latter kind. We use polynomial kernels to create combinations of features defined on LTAG derivation trees.

2.1 LTAG-based Feature Extraction

In order to create training data for the LTAG-based features, we convert the Penn Treebank phrase structure trees into LTAG derivations. First, we prune the Treebank parse tree using certain constraints. Then we decompose the pruned parse trees into a set of LTAG elementary trees and obtain a derivation tree. For each constituent in question, we extract features from the LTAG derivation tree. We combine these features with the standard features used for SRL and train an SVM classifier on the combined LTAG derivation plus SRL annotations from the PropBank corpus.

For the test data, we report on results using the gold-standard Treebank data, and in addition we also report results on automatically parsed data using the Charniak parser (Charniak, 2000) as provided by the CoNLL 2005 shared task. We did this for three reasons: (i) our results are directly comparable to those who have used the Charniak parses distributed with the CoNLL 2005 data-set; (ii) we avoid the possibility of a better parser identifying a larger number of argument constituents and thus leading to better results, which is orthogonal to the discriminative power of our proposed LTAG-based features; and (iii) the quality of LTAG derivation trees depends indirectly on the quality of head dependencies recovered by the parser and it is a well-known folklore result (see Table 3 in (McDonald et al., 2005)) that applying the head-percolation heuristics on parser output produces better dependencies when compared to dependencies directly recovered by the parser (whether the parser is an LTAG parser or a lexicalized PCFG parser).

2.1.1 Pruning Parse Trees

Given a parse tree, the pruning component identifies the predicate in the tree and then only admits those nodes that are sisters to the path from the predicate to the root. It is commonly used in the SRL community (cf. (Xue and Palmer, 2004)) and our experiments show that 91% of the SRL targets can be recovered despite this aggressive pruning. We make two enhancements to the pruned Propbank tree: we enrich the sister nodes with head information, a part-of-speech tag and word pair: $\langle t, w \rangle$ and PP nodes are expanded to include the NP complement of the PP (including head information). The target SRL node is still the PP. Figure 2 is a pruned parse tree for a sentence from the PropBank.

2.1.2 Decompositions of Parse Trees

After pruning, the pruned tree is decomposed around the predicate using standard head-percolation based heuristic rules[1] to convert a Treebank tree into an LTAG derivation tree. Figure 3 shows the resulting etrees after decomposition. Figure 4 is the derivation tree for the entire pruned tree. Each node in this derivation tree represents an etree in Figure 3. In our model we make an independence assumption that each SRL is assigned to each constituent independently, conditional only on the path from the *predicate etree* to the *argument etree* in the derivation tree. Different etree siblings in the LTAG derivation tree do not influence each other in our current models.

2.1.3 LTAG-based Features

We defined 5 LTAG feature categories: *predicate etree*-related features (**P** for short), *argument etree*-related features (**A**), *subcategorization*-related features (**S**), *topological relation*-related features (**R**), *intermediate etree*-related features (**I**). Since we consider up to 6 intermediate etrees between the predicate and the argument etree, we use *I*-1 to *I*-6 to represent these 6 intermediate trees respectively.

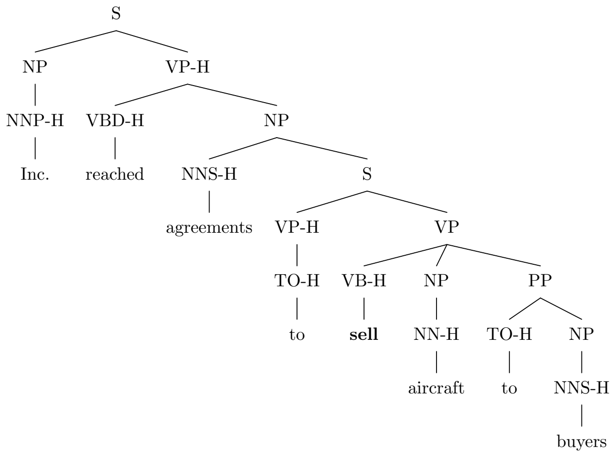


Figure 2: The pruned tree for the sentence "*Ports of Call Inc. reached agreements to* ***sell*** *its remaining seven aircraft to buyers that weren't disclosed.*"

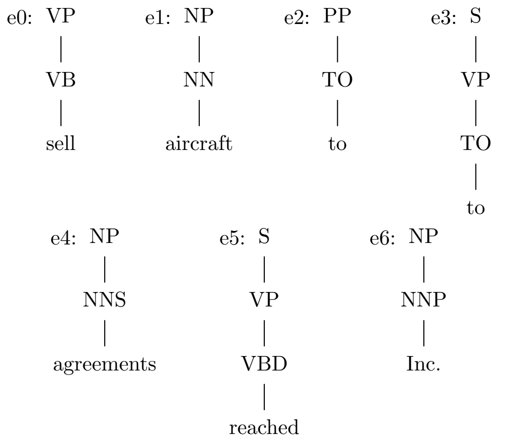


Figure 3: Elementary trees after decomposition of the pruned tree.

Category P: Predicate etree & its variants *Predicate etree* is an etree with predicate, such as $e0$ in Figure 3. This new feature complements the predicate feature in the standard SRL feature set. One variant is to remove the predicate lemma. Another variant is a combination of *predicate tree w/o predicate lemma&POS* and *voice*. In addition, this variant combined with predicate lemma comprises another new feature. In the example, these three variants are *(VP(VB))* and *(VP)_active* and *(VP)_active_sell* respectively.

Category A: Argument etree & its variants Analogous to the predicate etree, the *argument etree* is an etree with the target constituent and its head. Similar

[1] using http://www.isi.edu/∼chiang/software/treep/treep.html

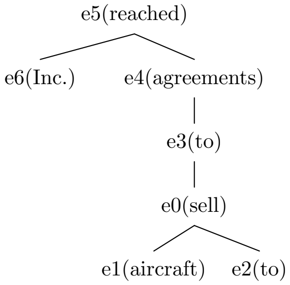


Figure 4: LTAG derivation tree for Figure 2.

to predicate etree related features, argument etree, argument etree with removal of head word, combination of *argument etree w/o head POS&head word* and *head Named Entity (NE) label (if any)* are considered. For example, in Figure 3, these 3 features for $e6$ are $e6$, *(NP(NNP))* and *(NP)_LOC* with head word "Inc." having NE label "LOC".

Category S: Index of current argument etree in subcat frame of predicate etree Sub-categorization is a standard feature that denotes the immediate expansion of the predicate's parent. For example, it is *V_NP_PP* for predicate *sell* in the given sentence. For argument etree $e1$ in Figure 3, the index feature value is 1 since it is the very first element in the (ordered) subcat sequence.

Category R:

Relation type between argument etree & predicate etree This feature is a combination of *position* and *modifying relation*. *Position* is a binary valued standard feature to describe if the argument is before or after the predicate in a parse tree. For each argument etree and intermediate etree, we consider three types of modifying relations they may have with the predicate etree: *modifying* (value 1), *modified* (value 2) and neither (value 3). From Figure 4, we can see $e1$ modifies $e0$ (predicate tree). So their modifying relation type value is 1; Combining this value with the position value, this feature for $e1$ is "1_*after*".

Attachment point of argument etree This feature describes where the argument etree is sister-adjoined/adjoined to the predicate etree, or the other way around. For $e1$ in the example, *VP* in the predicate tree is the attachment point.

Distance This feature is the number of intermediate etrees between argument etree and predicate etree in the derivation tree. In Figure 4, the distance from $e4$ to the predicate etree is 1 since only one intermediate etree $e3$ is between them.

Category I:

Intermediate etree related features *Intermediate etrees* are those etrees that are located between the predicate etree and argument etrees. The set of features we propose for each intermediate etree is quite similar to those for argument etrees except we do not consider the named-entity label for head words in this case.

Relation type of intermediate etree & predicate etree.

Attachment point of intermediate etree.

Distance between intermediate etree and predicate etree.

Up to 6 intermediate etrees are considered and the category I features are extracted for each of them (if they exist).

Each etree represents a linguistically meaningful fragment. The features of relation type, attachment point as well as the distance characterize the topological relations among the relevant etrees. In particular, the attachment point and distance features can explicitly capture important information hidden in the standard path feature. The intermediate tree related features can give richer contextual information between predicate tree and argument trees. We added the **subcat index** feature to be complementary to the *sub-cat* and *syntactic frame* features in the standard feature set.

3 Standard Feature Set

Our standard feature set is a combination of features proposed by (Gildea and Jurafsky, 2002), (Surdeanu et al., 2003; Pradhan et al., 2004; Pradhan et al., 2005b) and (Xue and Palmer, 2004). All features listed in Table 1 are used for argument classification in our baseline system; and features with asterisk are not used for argument identification[2]. We compare this baseline SRL system with a system that includes a combination of these features with the LTAG-based features. Our baseline uses *all* features that have been used in the state-of-the-art SRL systems and as our experimental results show, these standard features do indeed obtain state-of-the-art

[2] This is a standard idea in the SRL literature: removing features more useful for classification, e.g. named entity features, makes the classifier for identification more accurate.

Table 1: Standard features adopted by a typical SRL system. Features with asterisk $*$ are not used for argument identification.

Basic features from (Gildea and Jurafsky, 2002)
• **predicate lemma and voice**
• **phrase type and head word**
• **path from phrase to predicate** [1]
• **position:** phrase relative to predicate: before or after
• **sub-cat** records the immediate structure that expands from[2] predicate's parent
Additional features proposed by (Surdeanu et al. 2003; Pradhan et al., 2004, 2005)
• **predicate POS**
• **head word POS**
• **first/last word/POS**
• **POS of word immediately before/after phrase**
• **path length** [1]
• **LCA(Lowest Common Ancestor) path** from phrase to its lowest common ancestor with predicate
• **punctuation immediately before/after phrase**$*$
• **path trigrams**$*$: up to 9 are considered
• **head word named entity label** such as "PER, ORG, LOC"$*$
• **content word named entity label for PP parent node**$*$
Additional features proposed by (Xue and Palmer, 2004)
• **predicate_phrase type**
• **predicate_head word**
• **voice_position**
• **syntactic frame**$*$

[1] In Fig. 2 *NNS↑NP↓S↓VP↓VB* is the path from the constituent *NNS(agreements)* to the predicate *VB(sell)* and the path length is 4.

[2] This feature is different from the **frame** feature which usually refers to all the semantic participants for the particular predicate.

accuracy on the SRL task. We will show that adding LTAG-based features can improve the accuracy over this very strong baseline.

4 Experiments

4.1 Experimental Settings

Training data (PropBank Sections 2-21) and test data (PropBank Section 23) are taken from CoNLL-2005 shared task[3] All the necessary annotation information such as predicates, parse trees as well as Named Entity labels is part of the data. The argument set we consider is {A0, A1, A2, A3, A4, AM} where AM is a generalized annotation of all adjuncts such as AM-TMP, AM-LOC, etc., where PropBank function tags like TMP or LOC in AM-TMP, AM-LOC are ignored (a common setting for SRL, see (Xue and Palmer, 2004; Moschitti, 2004)). We chose these labels for our experiments because they have sufficient training/test data for the performance comparison and provide sufficient counts for accurate significance testing. However, we also provide the evaluation result on the test set for full CoNLL-2005 task (all argument types).

We use SVM-light[4] (Joachims, 1999) with a polynomial kernel (degree=3) as our binary classifier for argument classification. We applied a linear kernel to argument identification because the training cost of this phase is extremely computationally expensive. We use 30% of the training samples to fine tune the regularization parameter c and the loss-function cost parameter j for both stages of argument identification and classification. With parameter validation experiments, we set $c = 0.258$ and $j = 1$ for the argument identification learner and $c = 0.1$ and $j = 4$ for the argument classification learner.

The classification performance is evaluated using *Precision/Recall/F-score* (p/r/f) measures. We extracted all the gold labels of A0-A4 and AM with the argument constituent index from the original test data as the "gold output". When we evaluate, we contrast the output of our system with the gold output and calculate the p/r/f for each argument type.

Our evaluation criteria which is based on predicting the SRL for *constituents* in the parse tree is based on the evaluation used in (Toutanova et al., 2005). However, we also predict and evaluate those PropBank arguments which do not have a corresponding constituent in the gold parse tree or the automatic parse tree: the *missing constituent* case. We also evaluate *discontinuous* PropBank arguments using the notation used in the CoNLL-2005 data-set but we do not predict them. This is contrast with some previous studies where the problematic cases have been usually discarded or the largest constituents in the parse tree that almost capture the missing constituent cases are picked as being the correct answer. Note that, in addition to the constituent based evalu-

[3] http://www.lsi.upc.es/∼srlconll/.

[4] http://svmlight.joachims.org/

	Gold Standard		Charniak Parser	
	std	std+ltag	std	std+ltag
p(%)	95.66	96.79	87.71	89.11
r(%)	94.36	94.59	84.86	85.51
f(%)	95.00	**95.68**	86.26	**87.27**∗

Table 2: Argument identification results on test data

ation, in Section 4.4 we also provide the evaluation of our model on the CoNLL-2005 data-set.

Because the main focus of this work is to evaluate the impact of the LTAG-based features, we did not consider the frameset or a distribution over the entire argument set or apply any inference/constraints as a post-processing stage as most current SRL systems do. We focus our experiments on showing the value added by introducing LTAG-based features to the SRL task over and above what is currently used in SRL research.

4.2 Argument Identification

Table 2 shows results on argument identification (a binary classification of constituents into argument or non-argument). To fully evaluate the influence of the LTAG-based features, we report the identification results on both Gold Standard parses and on Charniak parser output (Charniak, 2000)[5].

As we can see, after combing the LTAG-based features with the standard features, F-score increased from 95.00% to 95.68% with Gold-standard parses; and from 86.26% to 87.27% with the Charniak parses (a larger increase). We can see LTAG-based features help in argument identification for both cases. This result is better than (Xue and Palmer, 2004), and better on gold parses compared to (Toutanova et al., 2005; Punyakanok et al., 2005b).

4.3 Argument Classification

Based on the identification results, argument classification will assign the semantic roles to the argument candidates. For each argument of A0-A4 and AM, a "one-vs-all" SVM classifier is trained on both the standard feature set (std) and the augmented feature set (std+ltag). Table 3 shows the classification results on the Gold-standard parses with the gold argument identification; Table 4 and 5 show the classification results on the Charniak parser with the gold argument identification and the automatic argument identification respectively. Scores for multi-class SRL are calculated based on the total number of correctly predicted labels, total number of gold labels and the number of labels in our prediction for this argument set.

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	96.69	96.71	96.71	96.77
	96.70		**96.74**	
A1	93.82	93.30	97.30	94.87
	93.56		**96.07**	
A2	87.05	79.98	92.43	81.42
	83.37		**86.58**	
A3	94.44	68.79	97.69	73.41
	79.60		**83.33**	
A4	96.55	82.35	94.11	78.43
	88.89		85.56	
AM	98.41	96.61	98.67	97.88
	97.50		**98.27**	
multi-class	95.35	93.62	97.15	94.70
	94.48		**95.91**	

Table 3: Argument classification results on Gold-standard parses with gold argument boundaries

4.4 Discussion

From the results shown in the tables, we can see that by adding the LTAG-based features, the overall performance of the systems is improved both for argument identification and for argument classification.

Table 3 and 4 show that with the gold argument identification, the classification for each class in {A0, A1, A2, A3, AM} consistently benefit from LTAG-based features. Especially for A3, LTAG-based features lead to more than 3 percent improvement. But for A4 arguments, the performance drops 3 percent in both cases. As we noticed in Table 5, which presents the argument classification results on Charniak parser output with the automatic argument identification, the prediction accuracy for classes A0, A1, A3, A4 and AM is improved, but drops a little for A2.

In addition, we also evaluated our feature set on the full CoNLL 2005 shared task. The over-

[5] We use the parses supplied with the CoNLL-2005 shared task for reasons of comparison.

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	96.04	92.92	96.07	92.92
	94.46		**94.47**	
A1	90.64	85.71	94.64	86.67
	88.11		**90.48**	
A2	84.46	75.72	89.26	75.22
	79.85		**81.64**	
A3	87.50	62.02	87.10	68.35
	72.59		**76.60**	
A4	90.00	79.12	90.54	73.62
	84.21		81.21	
AM	95.14	85.54	96.60	86.51
	90.09		**91.27**	
multi-	93.25	86.45	94.71	87.15
class	89.72		**90.77**	

Table 4: Argument classification results on Charniak parser output with gold argument boundaries

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	86.50	86.18	88.17	87.70
	86.34		**87.93**∗	
A1	78.73	83.82	88.78	85.22
	81.19		**86.97**∗	
A2	85.40	73.93	83.11	75.42
	79.25		79.08	
A3	85.71	60.76	85.71	68.35
	71.11		**76.06**∗	
A4	84.52	78.02	89.47	74.72
	81.15		**81.43**	
AM	80.47	82.11	83.87	81.54
	81.29		**82.69**∗	
multi-	81.79	82.90	86.04	84.47
class	82.34		**85.25**∗	

Table 5: Argument classification results on Charniak parser output with automatic argument boundaries

all performance using LTAG features increased from 74.41% to 75.31% in terms of F-score on the full argument set. Our accuracy is most closely comparable to the 78.63% accuracy achieved on the full task by (Pradhan et al., 2005a). However, (Pradhan et al., 2005a) uses some additional information since it deals with incorrect parser output by using multiple parsers. The 79.44% accuracy obtained by the top system in CoNLL 2005 (Punyakanok et al., 2005a) is not directly comparable since their system used the more accurate n-best parser output of (Charniak and Johnson, 2005). In addition their system also used global inference. Our focus in this paper was to propose *new LTAG features* and to evaluate impact of these features on the SRL task.

We also compared our proposed feature set against predicate/argument features (PAF) proposed by (Moschitti, 2004). We conducted an experiment using *SVM-light-TK-1.2 toolkit*[6]. The PAF tree kernel is combined with the standard feature vectors by a linear operator. With settings of Table 5, its multi-class performance (p/r/f)% is 83.09/80.18/81.61 with linear kernel and 85.36/81.79/83.53 with polynomial kernel (degree=3) over *std* feature vectors.

4.5 Significance Testing

To assess the statistical significance of the improvements in accuracy we did a two-tailed significance test on the results of both Table 2 and 5 where Charniak's parser outputs were used. We chose *SIGF*[7], which is an implementation of a computer-intensive, stratified approximate-randomization test (Yeh, 2000). The statistical difference is assessed on SRL identification, classification for each class (A0-A4, AM) and the full SRL task (overall performance). In Table 2 and 5, we labeled numbers under *std+ltag* that are statistically significantly better from those under *std* with asterisk. The significance tests show that for identification and full SRL task, the improvements are statistically significant with p value of 0.013 and 0.0001 at a confidence level of 95%. The significance test on each class shows that the improvement by adding LTAG-based features is statistically significant for class A0, A1, A3 and AM. Even though in Table 5 the performance of A2 appears to be worse it is not significantly so, and A4 is not significantly better. In comparison, the performance of PAF did not show significantly better than *std* with p value of 0.593 at the same confidence level of 95%.

[6] http://ai-nlp.info.uniroma2.it/moschitti/TK1.2-software/Tree-Kernel.htm

[7] http://www.coli.uni-saarland.de/∼pado/sigf/index.html

	full	**full−P**	**full−R**	**full−S**	**full−A**	**full−I**	**std**
id	90.5	90.6	90.0	90.5	90.5	90.1	89.6
A0	84.5	84.3	84.6	84.5	84.3	83.5	84.2
A1	89.8	90.1	89.4	89.3	89.6	89.3	88.9
A2	84.2	84.2	84.0	83.7	83.6	83.6	84.9
A3	76.7	80.7	75.1	76.0	75.6	76.7	78.6
A4	80.0	83.3	80.0	79.6	80.0	80.0	79.2
AM	82.8	83.3	82.9	82.8	82.6	83.1	82.4

Table 6: Impact of each LTAG feature category (**P, R, S, A, I** defined in Section 2.1.3) on argument classification and identification on CoNLL-2005 development set (WSJ Section 24). **full** denotes the full feature set, and we use $-\alpha$ to denote removal of a feature category of type α. For example, **full−P** is the feature set obtained by removing all **P** category features. **std** denotes the standard feature set.

5 Analysis of the LTAG-based features

We analyzed the drop in performance when a particular type of LTAG feature category is removed from the full set of LTAG features (we use the broad categories **P, R, S, A, I** as defined in Section 2.1.3). Table 6 shows how much performance is lost (or gained) when a particular type of LTAG feature is dropped from the full set.

These experiments were done on the development set from CoNLL-2005 shared task, using the provided Charniak parses. All the SVM models were trained using a polynomial kernel with degree 3. It is clear that the **S, A, I** category features help in most cases and **P** category features hurt in most cases, including argument identification. It is also worth noting that the **R** and **I** category features help most for identification. This vindicates the use of LTAG derivations as a way to generalize long paths in the parse tree between the predicate and argument. Although it seems LTAG features have negative impact on prediction of A3 arguments on this development set, dropping the **P** category features can actually improve performance over the standard feature set. In contrast, for the prediction of A2 arguments, none of the LTAG feature categories seem to help.

Note that since we use a polynomial kernel in the full set, we cannot rule out the possibility that a feature that improves performance when dropped may still be helpful when combined in a non-linear kernel with features from other categories. However, this analysis on the development set does indicate that overall performance may be improved by dropping the **P** feature category. We plan to examine this effect in future work.

6 Related Work

There has been some previous work in SRL that uses LTAG-based decomposition of the parse tree. (Chen and Rambow, 2003) use LTAG-based decomposition of parse trees (as is typically done for statistical LTAG parsing) for SRL. Instead of extracting a typical "standard" path feature from the derived tree, (Chen and Rambow, 2003) uses the path within the elementary tree from the predicate to the constituent argument. Under this frame, they only recover semantic roles for those constituents that are localized within a single etree for the predicate, ignoring cases that occur outside the etree. As stated in their paper, "as a consequence, adjunct semantic roles (ARGM's) are basically absent from our test corpus"; and around 13% complement semantic roles cannot be found in etrees in the gold parses. In contrast, we recover all SRLs by exploiting more general paths in the LTAG derivation tree. A similar drawback can be found in (Gildea and Hockenmaier, 2003) where a **parse tree path** was defined in terms of Combinatory Categorial Grammar (CCG) types using grammatical relations between predicate and arguments. The two relations they defined can only capture 77% arguments in Propbank and they had to use a standard path feature as a replacement when the defined relations cannot be found in CCG derivation trees. In our framework, we use intermediate sub-structures from LTAG derivations to capture these relations instead of bypassing this issue.

Compared to (Liu and Sarkar, 2006), we have used a more sophisticated learning algorithm and a richer set of syntactic LTAG-based features in this task. In particular, in this paper we built a strong baseline system using a standard set of features and did a thorough comparison between this strong baseline and our proposed system with LTAG-based features. The experiments in (Liu and Sarkar, 2006) were conducted on gold parses and it failed to show any improvements after adding LTAG-based features. Our experimental results show that LTAG-based features can help improve the performance of SRL systems. While (Liu and Sarkar, 2006) propose some new features for SRL based on LTAG derivations, we propose several novel features and in addition they do not show that their features are useful for SRL.

Our approach shares similar motivations with the approach in (Shen and Joshi, 2005) which uses PropBank information to recover an LTAG treebank as if it were hidden data underlying the Penn Treebank. However their goal was to extract an LTAG grammar using PropBank information from the Treebank, and *not* the SRL task.

Features extracted from LTAG derivations are different and provide distinct information when compared to predicate-argument features (PAF) or subcategorization features (SCF) used in (Moschitti, 2004) or even the later use of argument spanning trees (AST) in the same framework. The *adjunction* operation of LTAG and the extended domain of locality is not captured by those features as we have explained in detail in Section 2.

7 Conclusion and Future Work

In this paper we show that LTAG-based features improve on the best known set of features used in current SRL prediction systems: the F-score for argument identification increased from 86.26% to 87.27% and from 82.34% to 85.25% for the SRL task. The analysis of the impact of each LTAG feature category shows that the intermediate etrees are important for the improvement. In future work we plan to explore the impact that different types of LTAG derivation trees have on this SRL task, and explore the use of tree kernels defined over the LTAG derivation tree. LTAG derivation tree kernels were previously used for parse re-ranking by (Shen et al., 2003). Our work also provides motivation to do SRL and LTAG parsing simultaneously.

Acknowledgements

This research was partially supported by NSERC, Canada (RGPIN: 264905). We would like to thank Aravind Joshi, Libin Shen, and the anonymous reviewers for their comments.

References

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL-2005*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL-2000*.

J. Chen and O. Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *EMNLP-2003*.

J. Chen and K. Vijay-Shanker. 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proc. of the 6th International Workshop on Parsing Technologies (IWPT-2000), Italy*.

D. Chiang. 2000. Statistical parsing with an automatically extracted tree adjoining grammars. In *ACL-2000*.

D. Gildea and J. Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *EMNLP-2003*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 58(3):245–288.

D. Gildea and M. Palmer. 2002. The necessity of parsing for predicate argument recognition. In *ACL-2002*.

K. Hacioglu, S. Pradhan, W. Ward, J. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *CoNLL-2004 Shared Task*.

K. Hacioglu. 2004. Semantic role labeling using dependency trees. In *COLING-2004*.

T. Joachims. 1999. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Machines*.

A. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. *Handbook of Formal Languages*, 3.

Y. Liu and A. Sarkar. 2006. Using LTAG-Based Features for Semantic Role Labeling. In *TAG+8-2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *ACL-2005*.

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *ACL-2004*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

S. Pradhan, W. Ward, K. Hacioglu, , J. H. Martin, and D. Jurafsky. 2004. Shallow Semantic Parsing Using Support Vector Machines. In *HLT-NAACL-2004*.

S. Pradhan, K. Hacioglu, W. Ward, J. Martin, and D. Jurafsky. 2005a. Semantic role chunking combining complementary syntactic views. In *CoNLL-2005 Shared Task*.

S. Pradhan, W. Ward, K. Hacioglu, , J. H. Martin, and D. Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *ACL-2005*.

V. Punyakanok, D. Roth, and W. Yih. 2005a. Generalized inference with multiple semantic role labeling systems (shared task paper). In *CoNLL-2005*.

V. Punyakanok, D. Roth, and W. Yih. 2005b. The necessity of syntactic parsing for semantic role labeling. In *IJCAI-2005*.

L. Shen and A. Joshi. 2005. Building an LTAG Treebank. Technical Report Technical Report MS-CIS-05-15,5, CIS Department, University of Pennsylvania.

L. Shen, A. Sarkar, and A. Joshi. 2003. Using LTAG based features in parse reranking. In *EMNLP-2003*.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *ACL-2003*.

K. Toutanova, A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *ACL-2005*.

F. Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *EMNLP-2004*.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING-2000*.

Japanese Dependency Analysis Using the Ancestor-Descendant Relation

Akihiro Tamura†* **Hiroya Takamura**†† **Manabu Okumura**††
† Common Platform Software Research Laboratories NEC Corporation
a-tamura@ah.jp.nec.com
†† Precision and Intelligence Laboratory, Tokyo Institute of Technology, Japan
{takamura,oku}@pi.titech.ac.jp

Abstract

We propose a novel method for Japanese dependency analysis, which is usually reduced to the construction of a dependency tree. In deterministic approaches to this task, dependency trees are constructed by series of actions of attaching a bunsetsu chunk to one of the nodes in the tree being constructed. Conventional techniques select the node based on whether the new bunsetsu chunk and each node in the trees are in a parent-child relation or not. However, tree structures include relations between two nodes other than the parent-child relation. Therefore, we use ancestor-descendant relations in addition to parent-child relations, so that the added redundancy helps errors be corrected. Experimental results show that the proposed method achieves higher accuracy.

1 Introduction

Japanese dependency analysis has been recognized as one of the basic techniques in Japanese processing. A number of techniques have been proposed for years. Japanese dependency is usually represented by the relation between phrasal units called 'bunsetsu' chunks, which are the smallest meaningful sequences consisting of an independent word and accompanying words (e.g., a noun and a particle). Hereafter, a 'chunk' means a bunsetsu chunk in this paper. The relation between two chunks has a direction from the modifier to the modifiee. All dependencies in a sentence are represented by a dependency tree, where a node indicates a chunk, and node B is the parent of node A when chunk B is the modifiee of chunk A. Figure 1 shows an example of a dependency tree. The task of Japanese dependency analysis is to find the modifiee for each chunk in a sentence. The task is usually regarded as construction of a dependency tree.

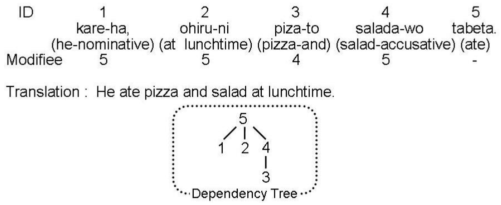

ID 1 2 3 4 5
kare-ha, ohiru-ni piza-to salada-wo tabeta.
(he-nominative) (at lunchtime) (pizza-and) (salad-accusative) (ate)
Modifiee 5 5 4 5 -
Translation : He ate pizza and salad at lunchtime.
5
1 2 4
3
Dependency Tree

Figure 1: Example of a dependency tree

In primitive approaches, the probabilities of dependencies are given by manually constructed rules and the modifiee of each chunk is determined. However, those rule-based approaches have problems in coverage and consistency. Therefore, a number of statistical techniques using machine learning algorithms have recently been proposed. In most conventional statistical techniques, the probabilities of dependencies between two chunks are learned in the learning phase, and then the modifiee of each chunk is determined using the learned models in the analysis phase. In terms of dependency trees, the parent node of each node is determined based on the likeliness of parent-child relations between two nodes.

We here take notice of the characteristics of dependencies which cannot be captured well only by

[*] Akihiro Tamura belonged to Tokyo Institute of Technology when this work was done.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 600–609, Prague, June 2007. ©2007 Association for Computational Linguistics

the parent-child relation. Consider, for example, Figure 1. In Figure 1, ID 3(pizza-and) and ID 4(salad-accusative) are in a parallel structure. In the structure, node 4 is a child of node 5(ate), but node 3 is not a child of 5, although 3 and 4 are both foods and should share a tendency of being subcategorized by the verb "eat". A number of conventional models use the pair of 3(pizza-and) and 5(ate) as a negative instance because 3 does not modify 5. Consequently, those models cannot learn and use the subcategorization preference of verbs well in the parallel structures.

We focus on ancestor-descendant relations to compensate for the weakness. Two nodes are in the ancestor-descendant relation when one of the two nodes is included in the path from the root node to the other node. The upper node of the two nodes is called an 'ancestor node' and the lower node a 'descendant node'. When the ancestor-descendant relation is used, both of the above two instances for nodes 3 and 4 can be considered as positive instances. Therefore, it is expected that the ancestor-descendant relation helps the algorithm capture the characteristics that cannot be captured well by the parent-child relation.

We aim to improve the performance of Japanese dependency analysis by taking the ancestor-descendant relation into account. In exploiting ancestor-descendant information, it came to us that redundant information is effectively utilized in a coding problem in communications (Mackay, 2003). Therefore, we propose a method in which the problem of determining the modifiee of a chunk is regarded as a kind of a coding problem: dependency is expressed as a sequence of values, each of which denotes whether a parent-child relation or an ancestor-descendant relation holds between two chunks.

In Section 2, we present the related work. In Section 3, we explain our method. In Section 4, we describe our experiments and their results, where we show the effectiveness of the proposed method. In Section 5, we discuss the results of the experiments. Finally, we describe the summary of this paper and the future work in Section 6.

2 Conventional Statistical Methods for Japanese Dependency Analysis

First, we describe general formulation of the probability model for dependency analysis. We denote a sequence of chunks, "$b_1, b_2, ..., b_m$", by B, and a sequence of dependency patterns, "$Dep(1), Dep(2), ..., Dep(m)$", by D, where $Dep(i) = j$ means that b_i modifies b_j. Given the sequence B of chunks as an input, dependency analysis is defined as the problem of finding the sequence D of the dependency patterns that maximizes the conditional probability $P(D \mid B)$. A number of the conventional methods assume that dependency probabilities are independent of each other and approximate $P(D \mid B)$ with $\prod_{i=1}^{m-1} P(Dep(i) \mid B)$. $P(Dep(i) \mid B)$ is estimated using machine learning algorithms. For example, Haruno et al. (1999) used Decision Trees, Sekine (2000) used Maximum Entropy Models, Kudo and Matsumoto (2000) used Support Vector Machines.

Another notable method is Cascaded Chunking Model by Kudo and Matsumoto (2002). In their model, a sentence is parsed by series of the following processes: whether or not the current chunk modifies the following chunk is estimated, and if it is so, the two chunks are merged together. Sassano (2004) parsed a sentence efficiently using a stack. The stack controls the modifier being analyzed.

These conventional methods determine the modifiee of each chunk based on the likeliness of dependencies between two chunks (in terms of dependency tree, the likeliness of parent-child relations between two nodes). The difference between the conventional methods and the proposed method is that the proposed method determines the modifiees based on the likeliness of ancestor-descendant relations in addition to parent-child relations, while the conventional methods tried to capture characteristics that cannot be captured by parent-child relations, by adding ad-hoc features such as features of "the chunk modified by the candidate modifiee" to features of the candidate modifiee and the modifier. However, these methods do not deal with ancestor-descendant relations between two chunks directly, while our method uses that information directly. In Section 5, we empirically show that our method uses the ancestor-descendant relation more effectively than the conventional ones and explain that our method is justifiable in terms of a coding problem.

3 Proposed Method

The methods explained in this section construct a dependency tree by series of actions of attaching a node to one of the nodes in the trees being constructed. Hence, when the parent node of a certain node is being determined, it is required that the parent node should already be included in the tree being constructed. To satisfy the requirement, we note the characteristic of Japanese dependencies: dependencies are directed from left to right. (i.e., the parent node is closer to the end of a sentence than its child node). Therefore, our methods analyze a sentence backwards as in Sekine (2000) and Kudo and Matsumoto (2000). Consider, for example, Figure 1. First, our methods determine the parent node of ID 4(salad-accusative), and then that of ID 3(pizza-and) is determined. Next, the parent node of ID 2(at lunchtime), and finally, that of ID 1(he-nominative) is determined and dependencies in a sentence are identified. Please note that our methods are applicable only to dependency structures of languages that have a consistent head-direction like Japanese.

We explain three methods that are different in the information used in determining the modifiee of each chunk. In Section 3.1, we explain PARENT METHOD and ANCESTOR METHOD, which determine the modifiee of each chunk based on the likeliness of only one type of the relation. PARENT METHOD uses the parent-child relation, which is used in conventional Japanese dependency analysis. ANCESTOR METHOD is novel in that it uses the ancestor-descendant relation which has not been used in the existing methods. In Section 3.2, we explain our method, PARENT-ANCESTOR METHOD, which determines the modifiees based on the likeliness of both ancestor-descendant and parent-child relations.

When the modifiee is determined using the ancestor-descendant relation, it is necessary to take into account the relations with every node in the tree. Consider, for example, the case that the modifiee of ID 1(he-nominative) is determined in Figure 1. When using the parent-child relation, the modifiee can be determined based only on the relation between ID 1 and 5. On the other hand, when using the ancestor-descendant relation, the modifiee cannot be determined based only on the relation between ID 1 and 5. This is because if one of ID 2, 3 and 4 is the modifiee of ID 1, the relation between ID 1 and 5 is ancestor-descendant. ID 5 is determined as the modifiee of ID 1 only after the relations with each node of ID 2, 3 and 4 are recognized not to be ancestor-descendant. An elegant way to use the ancestor-descendant relation, which we propose in this paper, is to represent a dependency as a codeword where each bit indicates the relation with a node in the tree, and determine the modifiee based on the relations with every node in the tree (for details to the next section).

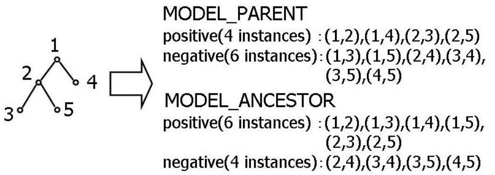
3.1 Methods with a single relation: PARENT METHOD and ANCESTOR METHOD

Figure 2 shows the pseudo code of the algorithm to construct a dependency tree using PARENT METHOD or ANCESTOR METHOD. As mentioned above, the two methods analyze a sentence backwards. We should note that $node_1$ to $node_n$ in the algorithm respectively correspond to the last chunk to the first chunk of a sentence. MODEL_PARENT($node_i$,$node_j$) indicates the prediction whether $node_j$ is the parent of $node_i$ or not, which is the output of the learned model. MODEL_ANCESTOR($node_i$,$node_j$) indicates the prediction whether $node_j$ is the ancestor of $node_i$ or not. $String\_output$ indicates the sequence of the $i-1$ predictions stored in step 3. The codeword denoted by $string[k]$ is the binary sequence given to the action that $node_i$ is attached to $node_k$. $Parent[node_i]$ indicates the node to which $node_i$ is attached, and *Dis* indicates a distance function. Thus, our method predicts the correct actions by measuring the distance between the codeword $string[k]$ and the predicted binary (later extended to real-valued) sequences $string\_output$. In other words, our method selects the action that is the closest to the outputs of the learned model.

Both models are learned from dependency trees given as training data as shown in Figure 3. Each relation is learned from ordered pairs of two nodes in the trees. However, our algorithm in Figure 2 targets at dependencies directed from left to right.

```
1:for $i = 1, 2, ..., n$ do
2:   for $j = 1, 2, ..., i-1$ do
3:      result_parent[$j$]=MODEL_PARENT($node_i$,$node_j$)
(in case of PARENT and PARENT-ANCESTOR METHOD)
3:      result_ancestor[$j$]=MODEL_ANCESTOR($node_i$,$node_j$)
(in case of ANCESTOR and PARENT-ANCESTOR METHOD)
4:   end
5:   Parent[$node_i$]=$argmin_k$ Dis($string[k], string\_output$)
6:end
```

Figure 2: Pseudo code of PARENT, ANCESTOR, and PARENT-ANCESTOR METHODS



MODEL_PARENT
positive(4 instances) : (1,2),(1,4),(2,3),(2,5)
negative(6 instances) : (1,3),(1,5),(2,4),(3,4),
(3,5),(4,5)
MODEL_ANCESTOR
positive(6 instances) : (1,2),(1,3),(1,4),(1,5),
(2,3),(2,5)
negative(4 instances) : (2,4),(3,4),(3,5),(4,5)

Figure 3: Example of training instances

Therefore, the instances with a right-to-left dependency are excluded from the training data. For example, the instance with $node4$ being the candidate parent (or ancestor) of $node1$ is excluded in Figure 3. MODEL_PARENT uses ordered pairs of a parent node and a child node as positive instances and the other ordered pairs as negative instances. MODEL_ANCESTOR uses ordered pairs of an ancestor node and a descendant node as positive instances and the other ordered pairs as negative instances. From the above description and Figure 3, the number of training instances used in learning MODEL_PARENT is the same as the number of training instances used in learning MODEL_ANCESTOR. However, the number of positive instances in learning MODEL_ANCESTOR is larger than in learning MODEL_PARENT because the set of parent-child relations is a subset of ancestor-descendant relations.

As mentioned above, the two methods analyze a sentence backwards. We should note that $node_1$ to $node_n$ in the algorithm respectively correspond to the last chunk to the first chunk of a sentence.

Next, we illustrate the process of determining the parent node of a certain node $node_m$(with Figures 4 and 5). Hereafter, $node_m$ is called a *target node*. The parent node is determined based on the likeliness of a relation; the parent-child and ancestor-descendant relation are used in PARENT METHOD and ANCESTOR METHOD respectively.

Our methods regard a dependency between the target node and its parent node as a set of relations between the target node and each node in the tree. Each relation corresponds to one bit, which becomes 1 if the relation holds, -1 otherwise. For example, a sequence $(-1, -1, -1, 1)$ represents that the parent of $node_5$ is $node_4$ in PARENT METHOD (Figure 4), since the relation holds only between nodes 4 and 5.

First, the learned model judges whether the target node and each node in the current tree are in a certain relation or not; PARENT METHOD uses MODEL_PARENT as the learned model and ANCESTOR METHOD uses MODEL_ANCESTOR. The sequence of the $m-1$ predictions by the learned model is stored in $string\_output$.

The codeword $string[k]$ is the binary (-1 or 1) sequence that is to be output when the target node is attached to the $node_k$. In Figures 4 and 5, the set of $string[k]$ (for $node_5$) is in the dashed square. For example, $string[2]$ in ANCESTOR METHOD (Figure 5) is $(1, 1, -1, -1)$ since nodes 1 and 2 are the ancestor of $node_5$ if $node_5$ is attached to $node_2$.

Next, among the set of $string[k]$, the codeword that is the closest to the $string\_output$ is selected. The target node is then attached to the node corresponding to the selected codeword. In Figure 4, the string[4], $(-1, -1, -1, 1)$, is selected and then $node_5$ is attached to $node_4$.

Japanese dependencies have the non-crossing constraint: dependencies do not cross one another. To satisfy the constraint, we remove the nodes that will break the non-crossing constraint from the candidates of a parent node in step 5 of the algorithm.

PARENT METHOD differs from conventional methods such as Sekine (2000) or Kudo and Matsumoto (2000), in the process of determining the parent node. These conventional methods select the node given by $argmax_j P(node_j \mid node_i)$ as the parent node of $node_i$, setting the beam width to 1. However, their processes are essentially the same as the process in PARENT METHOD.

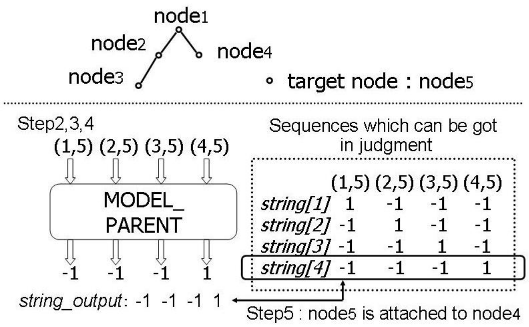


Figure 4: Analysis example using PARENT METHOD

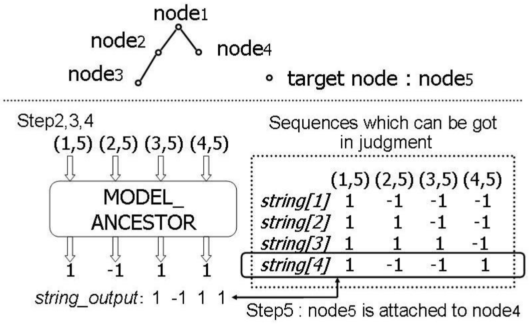


Figure 5: Analysis example using ANCESTOR METHOD

3.2 Proposed method: PARENT-ANCESTOR METHOD

The proposed method determines the parent node of a target node based on the likeliness of ancestor-descendant relations in addition to parent-child relations. The use of ancestor-descendant relations makes it possible to capture the characteristics which cannot be captured by parent-child relations alone. The pseudo code of the proposed method, PARENT-ANCESTOR METHOD, is shown in Figure 2. MODEL_PARENT and MODEL_ANCESTOR are learned as described in Section 3.1. $String\_output$ is the concatenation of the predictions by both MODEL_PARENT and MODEL_ANCESTOR. In addition, $string[k]$ is provided based not only on parent-child relations but also on ancestor-descendant relations. An analysis example using PARENT-ANCESTOR METHOD is shown in Figure 6.

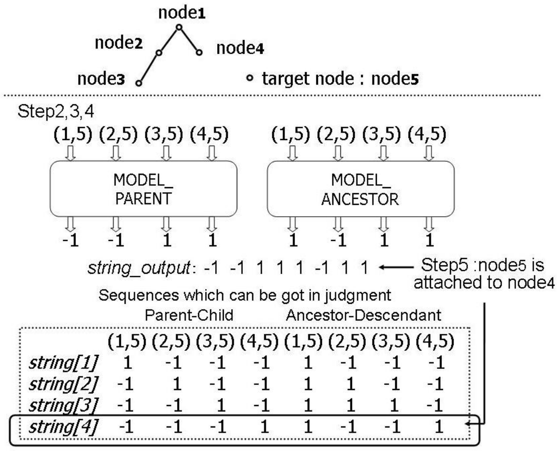

node1
node2
node4
node3
target node : node5
Step2,3,4
(1,5) (2,5) (3,5) (4,5)
(1,5) (2,5) (3,5) (4,5)
MODEL_ PARENT
MODEL_ ANCESTOR
-1 -1 1 1
1 -1 1 1
string_output: -1 -1 1 1 1 -1 1 1
Step5 :node5 is attached to node4
Sequences which can be got in judgment
Parent-Child
Ancestor-Descendant
(1,5) (2,5) (3,5) (4,5) (1,5) (2,5) (3,5) (4,5)
string[1] 1 -1 -1 -1 1 -1 -1 -1
string[2] -1 1 -1 -1 1 1 -1 -1
string[3] -1 -1 1 -1 1 1 1 -1
string[4] -1 -1 -1 1 1 -1 -1 1

Figure 6: Analysis example using PARENT-ANCESTOR METHOD

4 Experiment

4.1 Experimental settings

We used Kyoto University text corpus (Version 2.0) (Kurohashi and Nagao, 1997) for training and test data. The articles on January 1st through 8th (7,958 sentences) were used as training data, and the articles on January 9th (1,246 sentences) as test data. The dataset is the same as in leading works (Sekine, 2000; Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2002; Sassano, 2004).

We used SVMs as the algorithm of learning and analyzing the relations between nodes. We used the third degree polynomial kernel function and set the soft margin parameter C to 1, which is exactly the same setting as in Kudo and Matsumoto (2002). We can obtain the real-valued score in step 3 of the algorithm, which is the output of the separating function. The score can be regarded as likeliness of the two nodes being in the parent-child (or the ancestor-descendant). Therefore, we used the sequence of the outputs of SVMs as $string\_output$, instead of converting the scores into binary values indicating whether a certain relation holds or not.

Two feature sets are used: static features and dynamic features. The static features used in the experiments are shown in Table 1. The features are the same as those used in Kudo and Matsumoto (2002). In Table 1, $HeadWord$ means the rightmost content word in the chunk whose part-of-speech is not a functional category. $FunctionalWord$ means the

Table 1: Static features used in experiments

Modifier / Modifiee	$Head\ Word$ (surface-form, POS, POS-subcategory, inflection-type, inflection-form), $Functional\ Word$ (surface-form, POS, POS-subcategory, inflection-type, inflection-form), brackets, quotation-marks, punctuation-marks, position in sentence (beginning, end)
Between two chunks	distance (1,2-5,6-), case-particles, brackets, quotation-marks, punctuation-parks

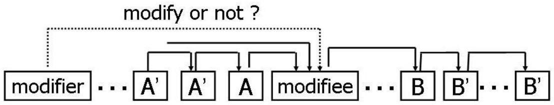


Figure 7: Dynamic features

rightmost functional word or the inflectional form of the rightmost predicate if there is no functional word in the chunk.

Next, we explain the dynamic features used in the experiments. Three types of dynamic features were used in Kudo and Matsumoto (2002): (A) the chunks modifying the current candidate modifiee, (B) the chunk modified by the current candidate modifiee, and (C) the chunks modifying the current candidate modifier. The type C is not available in the proposed method because the proposed method analyzes a sentence backwards unlike Kudo and Matsumoto (2002). Therefore, we did not use the type C. We used the type A' and B' which are recursive expansion of type A and B as the dynamic features (Figure 7). The form of functional words or inflection was used as a type A' feature and POS and POS-subcategory of $HeadWord$ as a type B' feature.

4.2 Experimental results

In this section, we show the effectiveness of the proposed method. First, we compare the three methods described in Section 3: PARENT METHOD, ANCESTOR METHOD, and PARENT-ANCESTOR METHOD. The results are shown in Table 2. Here, *dependency accuracy* is the percentage of correct dependencies (correct parent-child relations in trees in test data), and *sentence accuracy* is the percentage of the sentences in which all the modifiees are determined correctly (correctly constructed trees in test data).

Table 2: Result of dependency analysis using methods described in Section 3

Method	Dependency Accuracy	Sentence Accuracy
PARENT	88.95%	44.87%
ANCESTOR	87.64%	43.74%
PARENT-ANCESTOR	89.54%	47.38%

Table 3: Comparison to conventional methods

Feature	Method	Dependency Accuracy	Sentence Accuracy
Only static	Proposed method	88.88%	46.33%
	Kudo and Matsumoto (2002)	88.71%	45.19%
Static + Dynamic A,B	Proposed method	89.43%	47.94%
	Kudo and Matsumoto (2002)	89.19%	46.64%
Original	Proposed method	89.54%	47.38%
	Sekine (2000)	87.20%	40.76%
	Kudo and Matsumoto (2000)	89.09%	46.17%
	Kudo and Matsumoto (2002)	89.29%	47.53%
	Sassano (2004)	89.56%	48.35%
w/o Rich	Sassano (2004)	89.19%	47.05%
w/o Conj		89.41%	47.86%

Table 2 shows that PARENT-ANCESTOR METHOD is more accurate than the other two methods. In other words, the accuracy of dependency analysis improves by utilizing the redundant information. The improvement is statistically significant in the sign-test with 1% significance-level.

Next, we compare the proposed method with conventional methods. We compare the proposed method particularly with Kudo and Matsumoto (2002) with the same feature set. The reasons are that Cascaded Chunking Model proposed in Kudo and Matsumoto (2002) is used in a popular Japanese dependency analyzer, CaboCha [1], and the comparison can highlight the effectiveness of our approach because we can experiment under the same conditions (e.g., dataset, feature set, learning algorithm). A summary of the comparison is shown in Table 3.

Table 3 shows that the proposed method outperforms conventional methods except Sassano (2004)[2], while Sassano (2004) used richer features which are not used in the proposed method, such as features for conjunctive structures based on Kurohashi and Nagao (1994), features concerning the leftmost content word in the candidate modifiee. The comparison of the proposed method with Sassano (2004)'s method without the features of

[1] `http://chasen.org/˜taku/software/cabocha/`

[2] We have not tested the improvement statistically because we do not have access to the conventional methods.

Table 4: Accuracy of dependency analysis on parallel structures

	Parallel structures	Other than parallel structures
PARENT	74.18%	91.21%
ANCESTOR	73.24%	90.01%
PARENT-ANCESTOR	76.29%	91.63%

conjunctive structures (w/o Conj) and without the richer features derived from the words in chunks (w/o Rich) suggests that the proposed method is better than or comparable to Sassano (2004)'s method.

5 Discussion

5.1 Performance on parallel structures

As mentioned in Section 1, the ancestor-descendant relation is supposed to help to capture parallel structures. In this section, we discuss the performance of dependency analysis on parallel structures. Parallel structures such as those of nouns (e.g., Tom and Ken eat hamburgers.) and those of verbs (e.g., Tom eats hamburgers and drinks water.), are marked in Kyoto University text corpus. We investigate the accuracy of dependency analysis on parallel structures using the information.

Table 4 shows that the accuracy on parallel structures improves by adding the ancestor-descendant relation. The improvement is statistically significant in the sign-test with 1% significance-level. Table 4 also shows that error reduction rate on parallel structures by adding the ancestor-descendant relation is 8.3% and the rate on the others is 4.7%. These show that the ancestor-descendant relation work well especially for parallel structures.

In Table 4, the accuracy on parallel structures using PARENT METHOD is slightly better than that using ANCESTOR METHOD, while the difference is not statistically significant in the sign-test. It shows that the parent-child relation is also necessary for capturing the characteristics of parallel structures. Consider the following two instances in Figure 1 as an example: the ordered pair of ID 3(pizza-and) and ID 5(ate), and the ordered pair of ID 4(salad-accusative) and ID 5. In ANCESTOR METHOD, both instances are positive instances. On the other hand, only the ordered pair of ID 4 and ID 5 is a positive instance in PARENT METHOD. Hence, PARENT METHOD can learn appropriate case-particles in a modifier of a verb. For example, the particle which means "and" does not modify verbs. However, it is difficult for ANCESTOR METHOD to learn the characteristic. Therefore, both parent-child and ancestor-descendant relations are necessary for capturing parallel structures.

Table 5: Comparison between usages of the ancestor-descendant relation

	Dependency Accuracy	Sentence Accuracy
Feature	88.57%	44.71%
Model	88.88%	46.33%

5.2 Discussion on usages of the ancestor-descendant relation

In the proposed method, MODEL_ANCESTOR, which judges whether the relation between two nodes is ancestor-descendant or not, is prepared, and the information on the ancestor-descendant relation is directly utilized. On the other hand, conventional methods add the features regarding the ancestor or descendant chunk to capture the ancestor-descendant relation. In this section, we empirically show that the proposed method utilizes the information on the ancestor-descendant relation more effectively than conventional methods. The results in the previous sections could not show the effectiveness because MODEL_PARENT and MODEL_ANCESTOR in the proposed method use the features regarding the ancestor-descendant relation.

Table 5 shows the result of dependency analysis using two types of usages of the information on the ancestor-descendant relation. "Feature" indicates the conventional usage and "Model" indicates our usage. Please note that MODEL_PARENT and MODEL_ANCESTOR used in "Model" do not use the features regarding the ancestor-descendant relation. Table 5 shows that our usage is more effective than the conventional usage. This is because our usage takes advantage of redundancy in terms of a coding problem as described in the next section. Moreover, the learned features through the proposed method would include more information than ad-hoc features that were manually added.

5.3 Proposed method in terms of a coding problem

In a coding problem, redundancy is effectively utilized so that information can be transmitted more properly (Mackay, 2003). This idea is the same as the main point of the proposed method. In this section, we discuss the proposed method in terms of a coding problem.

In a coding problem, when encoding information, the redundant bits are attached so that the added redundancy helps errors be corrected. Moreover, the following fact is known (Mackay, 2003):

the error-correcting ability is higher when the distances between the codewords are longer. (1)

For example, consider the following three types of encodings: (A) two events are encoded respectively into the codewords -1 and 1 (the simplest encoding), (B) into the codewords $(-1, -1, 1)$ and $(1, 1, 1)$ (hamming distance:2), and (C) into the codewords $(-1, -1, -1)$ and $(1, 1, 1)$ (hamming distance:3). Please note that the hamming distance is defined as the number of bits that differ between two codewords. In (A), the correct information is not transmitted if a one-bit error occurs. In (B), if an error occurs in the third bit, the error can be corrected by assuming that the original codeword is closest to the received codeword. In (C), any one-bit error can be corrected. Thus, (B) has the higher error-correcting ability than (A), and (C) has the higher error-correcting ability than (B).

We explain the problem of determining the parent node of a target node in the proposed method in terms of the coding theory. A sequence of numbers corresponds to a codeword. It is assumed that the codeword which expresses the correct parent node of the target node is transmitted. The codeword is transmitted through the learned model through channels to the receiver. The receiver infers the parent node from the received sequence ($string\_output$) in consideration of the codewords that can be transmitted ($string[k]$). Therefore, error-correcting ability, the ability of correcting the errors in predictions in step 3, is dependent on the distances between the codewords ($string[k]$).

The codewords in PARENT-ANCESTOR METHOD are the concatenation of the bits based on both parent-child relations and ancestor-descendant relations. Consequently, the distances between codewords in PARENT-ANCESTOR METHOD are longer than those in PARENT METHOD or ANCESTOR METHOD. From (1), the error-correcting ability is expected to be higher. In terms of a coding problem, the proposed method exploits the essence of (1), and utilizes ancestor-descendant relations effectively.

We assume that every bit added as redundancy is correctly transmitted for the above-mentioned discussion. However, some of these added bits may be transmitted wrongly in the proposed method. In that case, the added redundancy may not help errors be corrected than cause an error. In the experiments of dependency analysis, the advantage prevails against the disadvantage because accuracy of each bit of the codeword is 94.5%, which is high value.

Discussion on applicability of existing codes

A number of approaches use Error Correcting Output Coding (ECOC) (Dietterich and Bakiri, 1995; Ghani, 2000) for solving multiclass classification problems as a coding problem. The approaches assign a unique n-bit codeword to each class, and then n classifiers are trained to predict each bit. The predicted class is the one whose codeword is closest to the codeword produced by the classifiers. The codewords in these approaches are designed to be well-separated from one another and have sufficient error-correcting ability (e.g., BCH code).

However, these existing codewords are not applicable to the proposed method. In the proposed method, we have two models respectively derived from the parent-child and ancestor-descendant relation, which can be interpreted in terms of both linguistic aspects and tree structures. If we use ECOC, however, pairs of nodes are divided into positive and negative instances arbitrarily. Since this division lacks linguistic or structural meaning, training instances will lose consistency and any proper model will not be obtained. Moreover, we have to prepare different models for each stage in tree construction, because the length of the codewords vary according to the number of nodes in the current tree.

Table 6: Result of dependency analysis using various distance functions

Distance Function	Method	Dependency Accuracy	Sentence Accuracy
Hamming	PARENT(n)	85.05%	35.35%
	PARENT(f)	85.48%	39.87%
	ANCESTOR(n)	87.54%	43.42%
	ANCESTOR(f)	86.97%	43.18%
	Proposed method(n)	88.36%	43.74%
	Proposed method(f)	88.45%	44.79%
Cosine / Euclidean	PARENT	88.95%	44.87%
	ANCESTOR	87.64%	43.74%
	Proposed method	89.54%	47.38%
Manhattan	PARENT(n)	88.74%	44.63%
	PARENT(f)	88.90%	44.79%
	ANCESTOR	87.64%	43.74%
	Proposed method	89.24%	46.89%

5.4 Influence of distance functions

In this section, we compare the performance of dependency analysis with various distance functions: hamming distance, euclidean distance, cosine distance, and manhattan distance. These distance functions between sequences X=“x_1 x_2 ... x_n” and Y=“y_1 y_2 ... y_n” are defined as follows:

- $\mathrm{Ham}(X,Y) = \sum_{i=1}^{n}(1-\delta(x_i,y_i))$,
- $\mathrm{Euc}(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$,
- $\mathrm{Cos}(X,Y) = 1 - \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$,
- $\mathrm{Man}(X,Y) = \sum_{i=1}^{n} \mid x_i - y_i \mid$.

In the hamming distance, $string\_output$ is converted to a binary sequence with their elements being of -1 or 1. The cosine distance is equivalent to the Euclidean distance under the condition that the absolute value of every component of $string[k]$ is 1.

The results of dependency analysis using these distance functions are shown in Table 6. In Table 6, ‘(n)’ means that the nearest chunk in a sentence is selected as the modifiee in order to break a tie, which happens when the number of sequences satisfying the condition in step 5 is two or more, while ‘(f)’ means that the furthest chunk is selected. If the results in case of (n) and (f) are the same, (n) and (f) are omitted and only one result is shown.

Table 6 shows that the proposed method outperforms PARENT METHOD and ANCESTOR METHOD in any distance functions. It means that the effectiveness of the proposed method does not depend on distance functions. The result using the hamming distance is much worse than using the other distance functions. It means that using the scores output by SVMs as the likeliness of a certain relation improves the accuracy. The results of (n) and (f) in the hamming distance are different. It is because the hamming distances are always positive integers and ties are more likely to happen. Table 6 also shows that the result of the cosine or the euclidean distance is better than that of the manhattan distance.

6 Conclusions

We proposed a novel method for Japanese dependency analysis, which determines the modifiee of each chunk based on the likeliness not only of the parent-child relation but also of the ancestor-descendant relation in a dependency tree. The ancestor-descendant relation makes it possible to capture the parallel structures in more depth. In terms of a coding theory, the proposed method boosts error-correcting ability by adding the redundant bits based on ancestor-descendant relations and increasing the distance between two codewords. Experimental results showed the effectiveness of the proposed method. In addition, the results showed that the proposed method outperforms conventional methods.

Future work includes the following. In this paper, we use the features proposed in Kudo and Matsumoto (2002). By extracting new features that are more suitable for the ancestor-descendant relation, we can further improve our method. The features used by Sassano (2004) are promising as well. We are also planning to apply the proposed method to other tasks which need to construct tree structures. For example, (zero-) anaphora resolution is considered as a good candidate task for application.

References

Thomas G. Dietterich and Ghulum Bakiri. 1995. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Rayid Ghani. 2000. Using Error-Correcting Codes For

Text Classification. In *Proc. of ICML-2000*, pages 303–310.

Masahiko Haruno, Satoshi Shirai, and Yoshifumi Ooyama. 1999. Using Decision Trees to Construct a Practical Parser. *Machine Learning*, 34:131–149.

Taku Kudo and Yuji Matsumoto. 2000. Japanese Dependency Analysis Based on Support Vector Machines. In *Proc. of EMNLP/VLC 2000*, pages 18–25.

Taku Kudo and Yuji Matsumoto. 2002. Japanese Dependency Analysis using Cascaded Chunking. In *Proc. of CoNLL 2002*, pages 63–69.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Sadao Kurohashi and Makoto Nagao. 1997. Kyoto University text corpus project. In *Proc. of ANLP*, pages 115–118, Japan.

David J. C. Mackay. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Manabu Sassano. 2004. Linear-Time Dependency Analysis for Japanese. In *Proc. of COLING 2004*, pages 8–14.

Satoshi Sekine. 2000. Japanese dependency analysis using a deterministic finite state transducer. In *Proc. of COLING 2000*, pages 761–767.

A Discriminative Learning Model for Coordinate Conjunctions

Masashi Shimbo[*]
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
shimbo@is.naist.jp

Kazuo Hara[*]
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
kazuo-h@is.naist.jp

Abstract

We propose a sequence-alignment based method for detecting and disambiguating coordinate conjunctions. In this method, averaged perceptron learning is used to adapt the substitution matrix to the training data drawn from the target language and domain. To reduce the cost of training data construction, our method accepts training examples in which complete word-by-word alignment labels are missing, but instead only the boundaries of coordinated conjuncts are marked. We report promising empirical results in detecting and disambiguating coordinated noun phrases in the GENIA corpus, despite a relatively small number of training examples and minimal features are employed.

1 Introduction

Coordination, along with prepositional phrase attachment, is a major source of syntactic ambiguity in natural language. Although only a small number of previous studies in natural language processing have dealt with coordinations, this does not mean disambiguating coordinations is easy and negligible; it still remains one of the difficulties for state-of-the-art parsers. in Charniak and Johnson's recent work (Charniak and Johnson, 2005), for instance, two of the features incorporated in their parse reranker are aimed specifically at resolving coordination ambiguities.

Previous work on coordinations includes (Agarwal and Boggess, 1992; Chantree et al., 2005; Kurohashi and Nagao, 1994; Nakov and Hearst, 2005; Okumura and Muraki, 1994; Resnik, 1999). Earlier studies (Agarwal and Boggess, 1992; Okumura and Muraki, 1994) attempted to find heuristic rules to disambiguate coordinations. More recent research are concerned with capturing structural similarity between conjuncts using thesauri and corpora (Chantree et al., 2005), or web-based statistics (Nakov and Hearst, 2005).

We identify three problems associated with the previous work.

1. Most of these studies evaluate the proposed heuristics against restricted forms of conjunctions. In some cases, they only deal with coordinations with exactly two conjuncts, leaving the generality of these heuristics unclear.

2. Most of these studies assume that the boundaries of coordinations are known in advance, which, in our opinion, is impractical.

3. The proposed heuristics and statistics capture many different aspects of coordination. However, it is not clear how they interact and how they can be combined.

To address these problems, we propose a new framework for detecting and disambiguating coordinate conjunctions. Being a discriminative learning model, it can incorporate a large number of overlapping features encoding various heuristics for coordination disambiguation. It thus provides a test bed for examining combined use of the proposed heuristics as well as new ones. As the weight on each feature is automatically tuned on the training data, assessing these weights allows us to evaluate the relative merit of individual features.

[*] Equal contribution.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 610–619, Prague, June 2007. ©2007 Association for Computational Linguistics

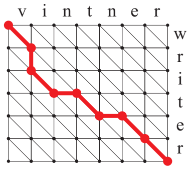


Figure 1: An alignment between ’writer’ and ’vintner,’ represented as a path in an edit graph

Our learning model is also designed to admit examples in which only the boundaries of coordinated conjuncts are marked, to reduce the cost of training data annotation.

The state space of our model resembles that of Kurohashi and Nagao’s Japanese coordination detection method (Kurohashi and Nagao, 1994). However, they considered only the decoding of coordinated phrases and did not address automatic parameter tuning.

2 Coordination disambiguation as sequence alignment

It is widely acknowledged that coordinate conjunctions often consist of two or more conjuncts having similar syntactic constructs. Our coordination detection model also follows this observation. To detect such similar constructs, we use the sequence alignment technique (Gusfield, 1997).

2.1 Sequence alignment

Sequence alignment is defined in terms of transformation of one sequence (string) into another through an *alignment*, or a series of edit operations. Each of the edit operations has an associated cost, and the cost of an alignment is defined as the total cost of edit operations involved in the alignment. The minimum cost alignment can be computed by dynamic programming in a state space called an *edit graph*, such as illustrated in Figure 1. In this graph, a complete path starting from the upper-left initial vertex and arriving at the lower-right terminal vertex constitutes a global alignment. Likewise, a partial path corresponds to a local alignment.

Sequence alignment can also be formulated with the *scores* of edit operations instead of their *costs*. In this case, the sequence alignment problem is that of finding a series of edit operations with the maximum score.

2.2 Edit graph for coordinate conjunctions

A fundamental difference between biological local sequence alignment and coordination detection is that the former deals with finding local homologies between two (or more) distinct sequences, whereas coordination detection is concerned with local similarities within a single sentence.

The maximal local alignment between two identical sequences is a trivial (global) alignment of identity transformation (the diagonal path in an edit graph). Coordination detection thus reduces to finding *off-diagonal* partial paths with the highest similarity score. Such paths never cross the diagonal, and we can limit our search space to the upper triangular part of the edit graph, as illustrated in Figure 2.

3 Automatic parameter tuning

Given a suitable substitution matrix, i.e., function from edit operations to scores, it is straightforward to find optimal alignments, or coordinate conjunctions in our task, by running the Viterbi algorithm in an edit graph.

In computational biology, there exist established substitution matrices (e.g., PAM and BLOSUM) built on a generative model of mutations and their associated probabilities.

Such convenient substitution matrices do not exist for coordination detection. Moreover, optimal score functions are likely to vary from one domain (or language) to another. Instead of designing a specific function for a single domain, we propose a general discriminative learning model in which the score function is a linear function of the *features* assigned to vertices and edges in the state space, and the weight of the features are automatically tuned for given gold standard data (training examples) drawn from the application domain. Designing heuristic rules for coordination detection, such as those proposed in previous studies, translates to the design of suitable features in our model.

Our learning method is an extension of Collins’s perceptron-based method for sequence labeling (Collins, 2002). However, a few incompatibilities exists between Collins’ sequence labeling method and edit graphs used for sequence alignment.

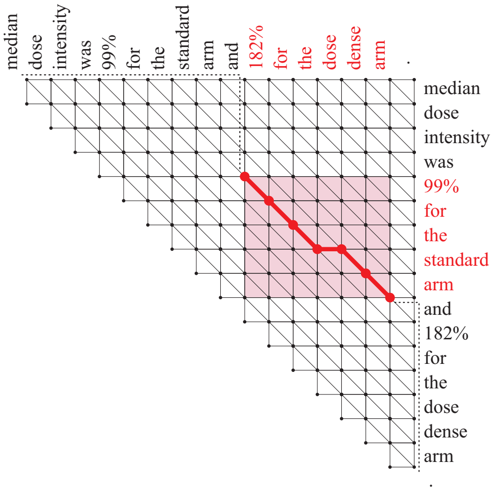


Figure 2: An edit graph for coordinate detection

1. Collins's method, like the linear-chain conditional random fields (CRFs) (Lafferty et al., 2001; Sha and Pereira, 2003), seeks for a complete path from the initial vertex to the terminal using the Viterbi algorithm. In an edit graph, on the other hand, coordinations are represented by partial paths. And we somehow need to complement the partial path to make a complete path.

2. A substitution matrix, which defines the score of edit operations, can be represented as a function of features defined on edges. But to deal with complex coordinations, a more expressive score function is sometimes desirable, so that scores can be computed not only on the basis of a single edit operation, but also on consecutive edit operations. Edit graphs are not designed to accommodate features for such a higher-order interaction of edit operations.

To reconcile these incompatibilities, we derive a more finer-grained model from the original edit graph. In presenting the description of our model below, we reserve the terminology 'vertex' and 'edge' for the original edit graph, and use 'node' and 'arc' for our new model, to avoid confusion.

3.1 State space for learning coordinate conjunctions

The new model is also based on the edit graph. In this model, we create a node for each triple (v, p, e), where v is a vertex in the original edit graph, $e \in \{Delete, Insert, Substitute\}$ is an admissible[1] edit operation at v, and $p \in \{Inside, Outside\}$ is a *polarity* denoting whether or not the edit operation e is involved in an alignment.

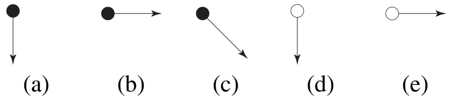


Figure 3: Five node types created for a vertex in an edit graph: (a) *Inside Delete*, (b) *Inside Insert*, (c) *Inside Substitute*, (d) *Outside Delete*, and (e) *Outside Insert*.

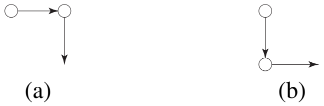


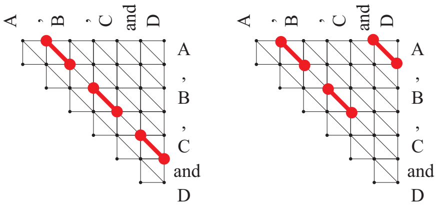
Figure 4: Series of edit operations with an equivalent net effect. (a) $(Insert, Delete)$, and (b) $(Delete, Insert)$. (b) is prohibited in our model.

For a node (v, p, e), we call the pair (p, e) its *type*. All five possible node types for a single vertex of an edit graph are shown in Figure 3. We disallow type $(Outside, Substitute)$, as it is difficult to attribute an intuitive meaning to substitution when two words are not aligned (i.e., *Outside*).

Arcs between nodes are built according to the transitions allowed in the original edit graph. To be precise, an arc between node (v_1, p_1, e_1) and node (v_2, p_2, e_2) is created if and only if the following three conditions are met. (i) Edit operations e_1 and e_2 are admissible at v_1 and v_2, respectively; (ii) the sink of the edge for e_1 at v_1 is v_2; and (iii) it is not the case with $p_1 = p_2$ and $(e_1, e_2) = (Delete, Insert)$.

Condition (iii) is introduced so as to disallow transition $(Delete, Insert)$ depicted in Figure 4(b). In contrast, the sequence $(Insert, Delete)$ (Figure 4(a)) is allowed. The net effects of these edit operation sequences are identical, in that they both skip one word each from the two sequences to be aligned. As a result, there is no use in discriminating between these two, and one of them, namely $(Delete, Insert)$, is prohibited.

[1] For a vertex v at the border of an edit graph, some edit operations are not applicable (e.g., *Insert* and *Substitute* at vertices on the right border in Figure 2); we say such operations are *inadmissible* at v. Otherwise, an edit operation is *admissible*.



(a) chainable (b) non-chainable

Figure 5: A coordination with four conjuncts represented as (a) chainable, and (b) non-chainable partial paths. We take (a) as the canonical representation.

3.2 Learning task

By the restriction of condition (iii) introduced above and the omission of (*Outside*, *Substitute*) from the node types, we can uniquely determine the complete path (from the initial node to the terminal node) that conjoins all the local alignments by *Outside* nodes (which corresponds to edges in the original edit graph). In Figure 2, the augmented *Outside* edges in this unique path are plotted as dotted lines for illustration.

Thus we obtain a complete path which is compatible with Collins's perceptron-based sequence learning method. The objective of the learning algorithms, which we will describe in Section 4, is to optimize the weight of features so that running the Viterbi algorithm will yield the same path as the gold standard.

Because a node in our state space corresponds to an edge in the original edit graph (see Figure 3), an arc in our state space is actually a pair of consecutive edges (or equivalently, edit operations) in the original graph. Hence our model is more expressive than the original edit graph in that the score function can have a term (feature) defined on a pair of edit operations instead of one.

3.3 More complex coordinations

Even if a coordination comprises three or more conjuncts, our model can handle them, as it can be represented as a set of pairwise local alignments that are *chainable* (Gusfield, 1997, Section 13.3). If pairwise local alignments are chainable, a unique complete path that conjoins all these alignments can be determined, allowing the same treatment as the case with two conjuncts.

For instance, a coordination with four conjuncts (A, B, C and D) can be decomposed into a set of pairwise alignments $\{(A,B),(B,C),(C,D)\}$ as depicted in Figure 5(a). This set of alignments are chainable and thus constitute the canonical encoding for this coordination; any other pairwise decomposition for these four conjuncts, like $\{(A,B),(B,C),(A,D)\}$ (Figure 5(b)), is not chainable.

Our model can handle multiple non-nested coordinations in a single sentence as well, as they can also be decomposed into chainable pairwise alignments. It cannot encode nested coordinations like (A, B, and (C and D)), however.

4 Algorithms

4.1 Reducing the cost of training data construction

Our learning method is supervised, meaning that it requires training data annotated with correct labels. Since a label in our problem is local alignments (or paths in an edit graph) representing coordinations, the training sentences have to be annotated with word-by-word alignments.

There are two reasons relaxing this requirement is desirable. First, it is expensive to construct such data. Second, there are coordinate conjunctions in which word-by-word correspondence is unclear even for humans. In Figure 2, for example, a word-by-word alignment of 'standard' with 'dense' is depicted, but it might be more natural to regard a word 'standard' as being aligned with two words 'dose dense' combined together.

Even if word-by-word alignment is uncertain, the boundaries of conjuncts are often obvious, and it is also much easier for human annotators to mark only the beginning and end of each conjunct. Thus we would like to allow for training examples in which only alignment boundaries are specified, instead of a full word-by-word alignment.

For these examples, conjunct boundaries corresponds to a rectangular region rather than a single path in an edit graph. The shaded box in Figure 2 illustrates the rectangular region determined by the boundaries of an alignment between the phrases "182% for the dose dense arm" and "99% for the standard arm." There are many possible alignment paths in this box, among which we do not know which one is correct (or even likely). To deal with

```
input:   Set of examples $S = \{(x_i, Y_i)\}$
         Iteration cutoff $T$
output:  Averaged weight vector $\bar{w}$
 1: $\bar{w} \leftarrow 0$; $w \leftarrow 0$
 2: for $t \leftarrow 1 \ldots T$ do
 3:     $\Delta w \leftarrow 0$
 4:     for each $(x_i, Y_i) \in S$ do
 5:         $y \leftarrow \arg\max_{y \in Y_i} w \cdot f(x_i, y)$
 6:         $y' \leftarrow \arg\max_{y \in A(x_i)} w \cdot f(x_i, y)$
 7:         $\Delta f \leftarrow f(x_i, y) - f(x_i, y')$
 8:         $\Delta w \leftarrow \Delta w + \Delta f$
 9:     end for
10:     if $\Delta w = 0$ then
11:         return $\bar{w}$
12:     end if
13:     $w \leftarrow w + \Delta w$
14:     $\bar{w} \leftarrow [(t-1)\bar{w} + w]/t$
15: end for
16: return $\bar{w}$
```

Figure 6: Path-based algorithm

```
input:   Set of examples $S = \{(x_i, Y_i)\}$
         Iteration cutoff $T$
output:  Averaged weight vector $\bar{w}$
$\bar{w} \leftarrow 0$; $w \leftarrow 0$
for each $(x_i, Y_i) \in S$ do
    $g_i \leftarrow (1/|Y_i|) \sum_{y \in Y_i} f(x_i, y)$
end for
for $t \leftarrow 1 \ldots T$ do
    $\Delta w \leftarrow 0$
    for each $(x_i, Y_i) \in S$ do
        $y' \leftarrow \arg\max_{y \in A(x_i)} w \cdot f(x_i, y)$
        Convert $y'$ into its box representation $Y'$
        $g' \leftarrow (1/|Y_i'|) \sum_{y \in Y_i'} f(x_i, y)$
        $\Delta f \leftarrow g_i - g'$
        $\Delta w \leftarrow \Delta w + \Delta f$
    end for
    if $\Delta w = 0$ then
        return $\bar{w}$
    end if
    $w \leftarrow w + \Delta w$
    $\bar{w} \leftarrow [(t-1)\bar{w} + w]/t$
end for
return $\bar{w}$
```

Figure 7: Box-based algorithm

this difficulty, we propose two simple heuristics we call the (i) path-based and (ii) box-based methods. As mentioned earlier, both of these methods are based on Collins's averaged-perceptron algorithm for sequence labeling (Collins, 2002).

4.2 Path-based method

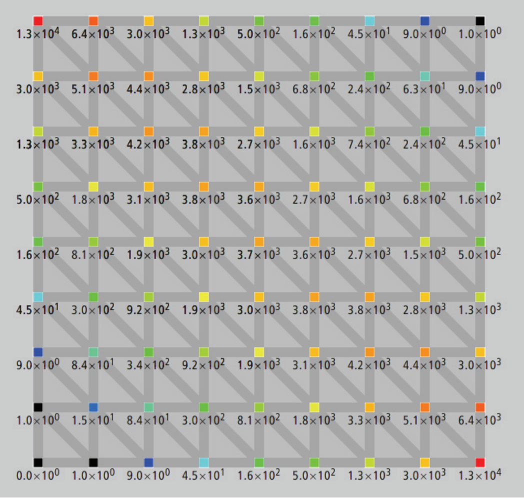
Our first method, which we call the "path-based" algorithm, is shown in Figure 6. We denote by $A(x)$ all possible alignments (paths) over x. The algorithm receives T, the maximum number of iterations, and a set of examples $S = \{(x_i, Y_i)\}$ as input, where x_i is a sentence (a sequence of words with their attributes, e.g., part-of-speech, lemma, prefixes, and suffixes) and $Y_i \subset A(x_i)$ is the set of admissible alignments (paths) for x_i. When a sentence is fully annotated with a word-by-word alignment y, $Y_i = \{y\}$ is a singleton set. In general boundary-only examples we described in Section 4.1, Y_i holds all possible alignments compatible with the marked range, or equivalently, paths that pass through the upper-left and lower-right corners of a rectangular region. Note that it is not necessary to explicitly enumerate all the member paths of Y_i; the set notation here is only for the sake of presentation.

The external function $f(x, y)$ returns a vector (called the *global feature vector* in (Sha and Pereira, 2003)) of the number of feature occurrences along the alignment path y. In the beginning (line 5 in the figure) of the inner loop, the target path (alignment) is recomputed with the current weight vector w. The arg max in lines 5 and 6 can be computed efficiently ($O(n^2)$, where n is the number of words in x) by running a pass of the Viterbi algorithm in the edit graph for x. The weight vector w varies between iterations, and so does the most likely alignment with respect to w. Hence the recomputation in line 5 is needed.

4.3 Box-based method

Our next method, called "box-based," is designed on the following heuristic. Given a rectangle region representing a local alignment (hence all nodes in the region are of polarity *Inside*) in an edit graph, we distribute feature weights in proportion to the probability of a node (or an arc) being passed by a path from the initial (upper left) node to the terminal (lower right) node of the rectangle. We assume paths are uniformly distributed.

Figure 8 displays an 8×8 sub-grid of an edit graph. The figure under each vertex shows the number of paths passing through the vertex. Vertices near the upper-left and the lower-right corner have a large frequency, and the frequency drops exponentially towards the top right corner and the bottom left corner, hence placing a strong bias on the paths near diagonals. This distribution fits our preference



1.3×10^4	6.4×10^3	3.0×10^3	1.3×10^3	5.0×10^2	1.6×10^2	4.5×10^1	9.0×10^0	1.0×10^0
3.0×10^3	5.1×10^3	4.4×10^3	2.8×10^3	1.5×10^3	6.8×10^2	2.4×10^2	6.3×10^1	9.0×10^0
1.3×10^3	3.3×10^3	4.2×10^3	3.8×10^3	2.7×10^3	1.6×10^3	7.4×10^2	2.4×10^2	4.5×10^1
5.0×10^2	1.8×10^3	3.1×10^3	3.8×10^3	3.6×10^3	2.7×10^3	1.6×10^3	6.8×10^2	1.6×10^2
1.6×10^2	8.1×10^2	1.9×10^3	3.0×10^3	3.7×10^3	3.6×10^3	2.7×10^3	1.5×10^3	5.0×10^2
4.5×10^1	3.0×10^2	9.2×10^2	1.9×10^3	3.0×10^3	3.8×10^3	3.8×10^3	2.8×10^3	1.3×10^3
9.0×10^0	8.4×10^1	3.4×10^2	9.2×10^2	1.9×10^3	3.1×10^3	4.2×10^3	4.4×10^3	3.0×10^3
1.0×10^0	1.5×10^1	8.4×10^1	3.0×10^2	8.1×10^2	1.8×10^3	3.3×10^3	5.1×10^3	6.4×10^3
0.0×10^0	1.0×10^0	9.0×10^0	4.5×10^1	1.6×10^2	5.0×10^2	1.3×10^3	3.0×10^3	1.3×10^4

Figure 8: Number of paths passing through the vertices of an 8×8 grid.

towards alignments with a larger number of substitutions.

The pseudo-code for the box-based algorithm is shown in Figure 7. For each example x_i and its possible target labels (alignments) Y_i, this algorithm first (line 3) computes and stores in the vector g_i the average number of feature occurrences in all possible target paths in Y_i. This quantity can be computed simply by summing over all nodes and edges feature occurrences multiplied by the pre-computed frequency of each nodes and arcs at which these features occur. analogously to the forward-backward algorithm. In each iteration, the algorithm scans every example (lines 7–13), computing the Viterbi path y' (line 8) according to the current weight vector w. Line 9 then converts y' to its box representation Y', by sequentially collapsing consecutive *Inside* nodes in y' as a box. For instance, let y' be the local alignment depicted as the bold line in Figure 2. The box Y' computed in line 9 for this y' is the shaded area in the figure. In parallel to the initialization step in line 3, we store in g' the average feature occurrences in Y' and update the current weight vector w by the difference between the target g_i and g'. These steps can be interpreted as a Viterbi approximation for computing the optimal set Y' of alignments directly.

5 Related work

5.1 Discriminative learning of edit distance

In our model, the state space of sequence alignment, or edit graph, is two-dimensional (which is actually three-dimensional if the dimension for labels is taken into account). This is contrastive to the one dimensional models used by Collins's perceptron-based sequence method (Collins, 2002) which our algorithms are based upon, and by the linear-chain CRFs.

McCallum et al. (McCallum et al., 2005) proposed a CRF tailored to learning string edit distance for the identity uncertainty problem. The state space in their work is two dimensional just like our model, but it is composed of two decoupled subspaces, each corresponding to 'match' and 'mismatch,' thus sharing only the initial state. It is not possible to make a transition from a state in the 'match' state space to the 'mismatch' space (and vice versa). As we can see from the decoupled state space, this method is based on global alignment rather than local alignment; it is not clear whether their method can identify local homologies in sequences. Our method uses a single state space in which both 'match (inside)' and 'mismatch (outside)' nodes co-exist and transition between them is permitted.

5.2 Inverse sequence alignment in computational biology

In computational biology, the estimation of a substitution matrix from data is called the *inverse sequence alignment* problem. Until recently, there have been a relatively small number of papers in this field despite a large body of literature in sequence alignment. Theoretical studies in the inverse sequence alignment include (Pachter and Sturmfels, 2004; Sun et al., 2004). Recently, CRFs have been applied for optimizing the substitution matrix in the context of global protein sequence alignment (Do et al., 2006).

6 Empirical evaluation

6.1 Dataset and Task

We used the GENIA Treebank beta corpus (Kim et al., 2003)[2] for evaluation of our methods. The cor-

[2] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA

pus consists of 500 parsed abstracts in Medline with a total of 4529 sentences.

Although the Penn Treebank Wall Street Journal (WSJ) is the de facto standard corpus for evaluating chunking and parsing performance, it lacks adequate structural information on coordinate conjunctions, and therefore does not serve our purpose. Many coordinations in the Penn Treebank are given a flat bracketing like (A, B, and C D), and thus we cannot tell which of ((A, B, and C) D) and ((A), (B), and (C D)) gives a correct alignment. The GENIA corpus, in contrast, distinguishes ((A, B, and C) D) and ((A), (B), and (C D)) explicitly, by providing more detailed bracketing. In addition, the corpus contains an explicit tag "COOD" for marking coordinations.

To avoid nested coordinations, which admittedly require techniques other than the one proposed in this paper, we selected from the GENIA corpus sentences in which the conjunction "and" occurs just once. After this operation, the number of sentences reduced to 1668, from which we further removed 32 that are not associated with the 'COOD' tag, and 3 more whose annotated tree structures contained obvious errors. Of the remaining 1633 sentences, 1061 were coordinated noun phrases annotated with NP-COOD tags, 226 coordinated verb phrases (VP-COOD), 142 coordinated adjective phrases (ADJP-COOD), and so on. Because the number of VP-COOD, ADJP-COOD, and other types of coordinated phrases are too small to make a meaningful benchmark, we focus on coordinated noun phrases in this experiment.

The task hence amounts to identifying coordinated NPs and their constituent conjuncts in the 1633 sentences, all of which contain a coordination marker "and" but only 1061 of which are actually coordinated NPs.

6.2 Baselines

We used several publicly available full parsers as baselines: (i) the Bikel parser (Bikel, 2005) version 0.9.9c with configuration file `bikel.properties` (denoted as Bikel/Bikel), (ii) the Bikel parser in the Collins parser emulation mode (using `collins.properties` file) (Bikel/Collins), and (iii) Charniak and Johnson's reranking parser (Charniak-Johnson) (Charniak and Johnson, 2005). We trained Bikel's parser and its Collins emulator with the GENIA corpus, WSJ, and the combination of the two. Charniak and Johnson's parser was used as distributed at Charniak's home page (and is WSJ trained).

Another baseline we used is chunkers based on linear-chain CRFs and the standard BIO labels. We trained two types of CRF-based chunkers by using different BIO sequences, one for the conjunct bracketing and the other for coordination bracketing. The chunkers were implemented with T. Kudo's CRF++ package version 0.45. We varied its regularization parameters C among $C \in \{0.01, 0.1, 1, 10, 100, 1000\}$, and the best results among these are reported below.

6.3 Features

Let $x = (x_1, \ldots, x_n)$ be a sentence, with its member x_k a vector of attributes for the kth word. The attributes include word surface, part-of-speech (POS), and suffixes, among others.

Table 1 summarizes (i) the features assigned to a node whose corresponding edge in the original edit graph for x is emanating from row i and column j, and (ii) the features assigned to the arcs (consisting of two edges in the original edit graph) whose joint (the vertex between the two edges) is a vertex at row i and column j.

We also tested the path-based and box-based methods, and the CRF chunkers both with and without the word and suffix features.

Although this is not a requirement of our model or algorithms, every feature we use in this experiment is binary; if the condition associated with a feature is satisfied, the feature takes a value of 1; otherwise, it is 0. A condition typically asks whether or not specific attributes match those at a current node, arc, or their neighbors.

We used the POS tags from the GENIA corpus as the POS attribute. The morphological features include 3- and 4-gram suffixes and indicators of whether a word includes capital letters, hyphens, and digits.

For the baseline CRF-based chunkers, we assign the word, POS (from GENIA), and the morphological features to nodes, and the POS features to edges. The feature set is identical to those used for our proposed methods, except for features defined on row-column combination (i.e., those defined over both i

Table 1: Features for the proposed methods

Substitute **(diagonal) nodes** $(*, \mathit{Substitute}, *)$	Indicators of the word, POS, and morphological attributes of x_i, x_j, (x_{i-1},x_i), (x_i,x_{i+1}), (x_{j-1},x_j), (x_j,x_{j+1}), and $(x_i,\ x_j)$, respectively combined with the type of the node.
	For each of the word, POS, and morphological attributes, an indicator of whether the respective attribute is identical in x_i and x_j, combined with the type of the node.
Delete **(vertical) nodes** $(*, \mathit{Delete}, *)$	Indicators of the word, POS, and morphological attributes of x_i, x_j, x_{j-1}, (x_{i-1},x_i), (x_i,x_{i+1}), and (x_{j-1},x_j), combined with the type of the node.
Insert **(horizontal) nodes** $(*, \mathit{Insert}, *)$	Indicators of the word, POS, and morphological attributes of x_i, x_{i-1}, x_j, (x_{i-1},x_i), (x_{j-1},x_j), and $(x_j,\ x_{j+1})$, combined with the type of the node.
Any arcs $(*,*,*) \rightarrow (*,*,*)$	Indicators of the POS attribute of x_i, x_{i-1}, x_j, x_{j-1}, (x_{i-2},x_{i-1}), (x_{i-1},x_i), (x_i,x_{i+1}), (x_{j-2},x_{j-1}), (x_{j-1},x_j), (x_j,x_{j+1}), (x_{i-1},x_{j-1}), (x_{i-1},x_j), (x_i,x_{j-1}) and (x_i,x_j), combined with the type pair of the arc.
Arcs between nodes of different polarity $(*, \mathit{Inside}, *) \rightarrow (*, \mathit{Outside}, *)$ and $(*, \mathit{Outside}, *) \rightarrow (*, \mathit{Inside}, *)$	Indicator of the distance $j - i$ between two words x_i and x_j, combined with the type pair of the arc.

and j in Table 1. The latter cannot be incorporated as a local features in chunkers based on linear chain.

For the Bikel (and its Collins emulation) parsers which accepts POS tags output by external taggers upon testing, we gave them the POS tags from the GENIA corpus, for fair comparison with the proposed methods and CRF-based chunkers.

6.4 Evaluation criteria

We employed two evaluation criteria: (i) correctness of the conjuncts output by the algorithm, and (ii) correctness of the range of coordinations as a whole.

For the correctness of conjuncts, we further use two evaluation criteria. The first evaluation method ("pairwise evaluation") is based on the decomposition of coordinations into the canonical set of pairwise alignments, as described in Section 3.3. After the set of pairwise alignments is obtained, each pairwise alignment is transformed into a box surrounded by their boundaries. Using these boxes, we evaluate precision, recall and F rates through the following definition. The precision measures how many of the boxes output by the algorithm exactly match those in the gold standard, and the recall rate is the percentage of boxes found by the algorithm. The F rate is the harmonic mean of the precision and the recall.

The second evaluation method ("chunk-based evaluation") for conjuncts is based on whether the algorithm correctly outputs the beginning and end of each conjunct, in the same manner as the chunking tasks. Here, we adopt the evaluation criteria for the CoNLL 99 NP bracketing task[3]; the precision equals how many of the NP conjuncts output by the algorithm are correct, and the recall is the percentage of NP conjuncts found by the algorithm.

Of these two evaluation methods for conjuncts, it is harder to obtain a higher pairwise evaluation score than the chunk-based evaluation. To be counted as a true positive in the pairwise evaluation, two consecutive chunks must be output correctly by the algorithm.

For the correctness of the coordination range, we check if both the start of the first coordinated conjunct and the end of the last conjunct in the gold match those output by the algorithm The reason we evaluate coordination range is to compare our proposed method with the full parsers trained on WSJ (but applied to GENIA). Although WSJ and GENIA differ in the way conjuncts are annotated, they are mostly identical on how the range of coordinations are annotated, and hence comparison is feasible in terms of coordination range. For the baseline parsers, we regard the bracketing directly surrounding the coordination marker "and" as their output.

In (Clegg and Shepherd, 2007), an F score of 75.5 is reported for the Bikel parser on coordination detection. Their evaluation is based on dependencies, which is different from our evaluation criteria which are all based on boundaries. Generally speaking, our evaluation criterion seems stricter, as exemplified in Figures 7 and 8 of Clegg and Shepherd's paper; in these figures, our evaluation criterion would result

[3] http://www.cnts.ua.ac.be/conll99/npb/

Table 2: Performance on conjunct bracketing. P: precision (%), R: recall (%), F: F rate.

	Pairwise evaluation			Chunk-based evaluation		
Method	P	R	F	P	R	F
Path-based method	61.4	56.2	58.7	70.9	66.9	68.9
Path-based method without word and suffix features	61.7	**58.8**	**60.2**	**71.2**	**69.7**	**70.5**
Box-based method	60.6	58.3	59.4	70.5	69.1	69.8
Box-based method without word and suffix features	59.5	58.3	58.9	69.7	69.5	69.6
Linear-chain CRF chunker (conjunct bracketing)	**62.6**	51.4	56.4	71.0	66.1	68.5
Bikel/Collins, trained with GENIA	50.0	48.6	49.3	65.0	64.2	64.6
Bikel/Bikel, trained with GENIA	50.1	47.8	49.0	63.9	61.3	62.6

Table 3: Performance on coordination bracketing. P: precision (%), R: recall (%), F: F rate.

Method	P	R	F
Path-based method	58.2	55.3	56.7
Path-based method without words and suffix features	57.7	**56.6**	**57.2**
Box-based method	55.6	54.4	55.0
Box-based method without words and suffix features	54.8	54.6	54.7
Linear-chain CRF chunker, trained with conjunct bracketing	43.9	46.7	45.3
Linear-chain CRF chunker, trained with coordination bracketing	**58.4**	51.0	54.5
Bikel/Collins, trained with GENIA	44.0	45.4	44.7
Bikel/Collins, trained with WSJ	42.3	43.2	42.7
Bikel/Collins, trained with GENIA+WSJ	43.3	45.1	44.1
Bikel/Bikel, trained with GENIA	44.8	45.4	45.1
Bikel/Bikel, trained with WSJ	40.7	41.5	41.1
Bikel/Bikel, trained with GENIA+WSJ	43.9	45.8	44.9
Charniak-Johnson reranking parser	48.3	45.2	46.7

in zero true positive, whereas their evaluation counts the dependency arc from ‘genes’ to ‘human’ as one true positive.

6.5 Results

The results of conjunct and coordination bracketing are shown in Tables 2 and 3, respectively. These are the results of a five-fold cross validation. We ran the proposed methods until convergence or the cutoff iteration of $T = 10000$, whichever comes first.

The path-based method (without words and suffixes) and box-based method (with full features) each achieved 2.0 and 1.3 point improvements over the CRF chunker in terms of the F score in conjunct identification (chunk-based evaluation), 3.8 and 3.0 point improvement in terms of pairwise evaluation, and 2.7 and 0.5 points in coordinate identification, respectively. Our methods also showed a performance considerably higher than the baseline parsers.

The performance of the path-based method was better when the word and suffix features were removed, while the box-based method and CRF chunkers performed better with these features.

7 Conclusions

We have proposed a new coordination learning and disambiguation method that can incorporate many different features, and automatically optimize their weights on training data.

In the experiment of Section 6, the proposed method obtained a performance superior to a linear-chain chunker and to the state-of-art full parsers.

We used only syntactic and morphological features, and did not use external similarity measures like thesauri and corpora, although they are reported to be effective for disambiguating coordinations. We note that it is easy to incorporate such external similarity measures as a feature in our model, thanks to its two-dimensional state space. The similarity of two words derived from an external knowledge base can be assigned to a *Substitute* node at a corresponding location in the state space in a straightforward manner. This is a topic we are currently working on.

We are also planning to reimplement our algorithms using CRFs instead of the averaged perceptron algorithm.

References

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of the 30th Annual Meeting of the Association for Computing Linguistics (ACL'92)*, pages 15–21.

Daniel M. Bikel. 2005. Multilingual statistical parsing engine version 0.9.9c. http://www.cis.upenn.edu/~dbikel/software.html.

Francis Chantree, Adam Kilgarriff, Anne de Roeck, and Alistair Willis. 2005. Disambiguating coordinations using word distribution information. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, Borovets, Bulgaria.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-2005)*.

Andrew B Clegg and Adrian J Shepherd. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8(24).

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

C. B. Do, S. S. Gross, and S. Batzoglou. 2006. CONTRAlign: discriminative training for protein sequence alignment. In *Proceedings of the Tenth Annual International Conference on Computational Molecular Biology (RECOMB 2006)*.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus: a semantically annotated corpus for biotextmining. *Bioinformatics*, 19(Suppl. 1):i180–i182.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20:507–534.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pages 282–289. Morgan Kaufmann.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*.

Preslav Nakov and Marti Hearst. 2005. Using the web as an implicit training set: application to structural ambiguity resolution. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language (HLT/EMNLP)*, pages 835–842, Vancouver.

Akitoshi Okumura and Kazunori Muraki. 1994. Symmetric pattern matching analysis for English coordinate structures. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 41–46.

Lior Pachter and Bernd Sturmfels. 2004. Parametric inference for biological sequence analysis. *Proceedings of the National Academy of Sciences of the USA*, 101(46):16138–16143.

Philip Resnik. 1999. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference North American Chapter of Association for Computational Linguistics (HLT-NAACL 2003)*, pages 213–220, Edmonton, Alberta, Canada. Association for Computational Linguistics.

Fangting Sun, David Fernández-Baca, and Wei Yu. 2004. Inverse parametric sequence alignment. *Journal of Algorithms*, 53:36–54.

Recovery of Empty Nodes in Parse Structures

Denis Filimonov[1]
[1]University of Maryland
College Park, MD 20742
den@cs.umd.edu

Mary P. Harper[1,2]
[2]Purdue University
West Lafayette, IN 47907
mharper@casl.umd.edu

Abstract

In this paper, we describe a new algorithm for recovering WH-trace empty nodes. Our approach combines a set of hand-written patterns together with a probabilistic model. Because the patterns heavily utilize regular expressions, the pertinent tree structures are covered using a limited number of patterns. The probabilistic model is essentially a probabilistic context-free grammar (PCFG) approach with the patterns acting as the terminals in production rules. We evaluate the algorithm's performance on gold trees and parser output using three different metrics. Our method compares favorably with state-of-the-art algorithms that recover WH-traces.

1 Introduction

In this paper, we describe a new algorithm for recovering WH-trace empty nodes in gold parse trees in the Penn Treebank and, more importantly, in automatically generated parses. This problem has only been investigated by a handful of researchers and yet it is important for a variety of applications, e.g., mapping parse trees to logical representations and structured representations for language modeling. For example, SuperARV language models (LMs) (Wang and Harper, 2002; Wang et al., 2003), which tightly integrate lexical features and syntactic constraints, have been found to significantly reduce word error in English speech recognition tasks. In order to generate SuperARV LM training, a state-of-the-art parser is used to parse training material and then a rule-based transformer converts the parses to the SuperARV representation. The transformer is quite accurate when operating on treebank parses; however, trees produced by the parser lack one important type of information – gaps, particularly WH-traces, which are important for more accurate extraction of the SuperARVs.

Approaches applied to the problem of empty node recovery fall into three categories. Dienes and Dubey (2003) recover empty nodes as a preprocessing step and pass strings with gaps to their parser. Their performance was comparable to (Johnson, 2002); however, they did not evaluate the impact of the gaps on parser performance. Collins (1999) directly incorporated wh-traces into his Model 3 parser, but he did not evaluate gap insertion accuracy directly. Most of the research belongs to the third category, i.e., post-processing of parser output. Johnson (2002) used corpus-induced patterns to insert gaps into both gold standard trees and parser output. Campbell (2004) developed a set of linguistically motivated hand-written rules for gap insertion. Machine learning methods were employed by (Higgins, 2003; Levy and Manning, 2004; Gabbard et al., 2006).

In this paper, we develop a probabilistic model that uses a set of patterns and tree matching to guide the insertion of WH-traces. We only insert traces of non-null WH-phrases, as they are most relevant for our goals. Our effort differs from the previous approaches in that we have developed an algorithm for the insertion of gaps that combines a small set of expressive patterns with a probabilistic grammar-based model.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 620–629, Prague, June 2007.

2 The Model

We have developed a set of tree-matching patterns that are applied to propagate a gap down a path in a parse tree. Pattern examples appear in Figure 1. Each pattern is designed to match a subtree (a root and one or more levels below that root) and used to guide the propagation of the trace into one or more nodes at the terminal level of the pattern (indicated using directed edges). Since tree-matching patterns are applied in a top-down fashion, multiple patterns can match the same subtree and allow alternative ways to propagate a gap. Hence, we have developed a probabilistic model to select among the alternative paths. We have created 24 patterns for WHNP traces, 16 for WHADVP, 18 for WHPP, and 11 for WHADJP.

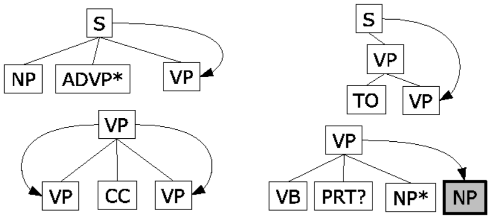


Figure 1: Examples of tree-matching patterns

Before describing our model, we first introduce some notation.

- T_{ij}^{N} is a tree dominating the string of words between positions i and j with N being the label of the root. We assume there are no unary chains like $N - X - ... - Y - N$ (which could be collapsed to a single node N) in the tree, so that T_{ij}^{N} uniquely describes the subtree.
- A gap location $g_{cd}^{ab,N}$ is represented as a tuple $(gaptype, ancstr(a, b, N), c, d)$, where $gaptype$ is the type of the gap, (e.g., $whnp$ for a WHNP trace), $ancstr(a, b, N)$ is the gap's nearest ancestor, with a and b being its span and N being its label, and c and d indicating where the gap can be inserted. Note that a gap's location is specified precisely when $c = d$. If the gap is yet to be inserted into its final location but will be inserted somewhere inside $ancstr(a, b, N)$, then we set $c = a$ and $d = b$.
- $ancstr(a, b, N)$ in the tuple for $g_{xy}^{ab,N}$ is the tree T_{ab}^{N}.
- $p(g_{xy}^{ab,N}|gaptype, T_{ij}^{N})$ is the probability that a gap of $gaptype$ is located between x and y, with a and b being the span of its ancestor, and $i \leq a \leq x \leq y \leq b \leq j$.

Given this notation, our model is tasked to identify the best location for the gap in a parse tree among the alternatives, i.e.,

$$\underset{x,a,b,N}{argmax}\ Pr(g_{xx}^{ab,N}|T, gaptype)$$

where $g_{xx}^{ab,N}$ represents a gap location in a tree, and $T = T_{ij}^{N}$ is the subtree of the parse tree whose root node is the nearest ancestor node dominating the WH-phrase, excluding the WH-node itself, and $gaptype$ is the type of the gap. In order to simplify the notation, we will omit the root labels N in T_{ij}^{N} and $g_{xy}^{ab,N}$, implying that they match where appropriate.

To guide this model, we utilize tree-matching patterns (see Figure 1), which are formally defined as functions:

$$ptrn : \mathcal{T} \times \mathcal{G} \rightarrow \mathbf{\Gamma} \cup \{none\}$$

where $\mathcal{T}$ is the space of parse trees, $\mathcal{G}$ is the space of gap types, and $\mathbf{\Gamma}$ is the space of gaps g_{cd}^{ab}, and $none$ is a special value representing failure to match[1]. The application of a pattern is defined as: $app(ptrn, \tau, gaptype) = ptrn(\tau, gaptype)$, where $\tau \in \mathcal{T}$ and $gaptype \in \mathcal{G}$. We define application of patterns as follows:

$$\begin{aligned}
app(ptrn, T_{ij}, gaptype) &\rightarrow g_{xy}^{ab} : i \leq a \leq x < y \leq b \leq j \\
app(ptrn, T_{ij}, gaptype) &\rightarrow g_{xx}^{ab} : i \leq a \leq x \leq b \leq j \\
app(ptrn, T_{ij}, gaptype) &\rightarrow none
\end{aligned}$$

Because patterns are uniquely associated with specific gap types, we will omit $gaptype$ to simplify the notation. Application is a function defined for every pair $(ptrn, T_{ij})$ with fixed $gaptype$. Patterns are applied to the root of T_{ij}, not to an arbitrary subtree.

Consider an example of pattern application shown in Figure 2. The tree contains a relative clause such that the WHNP-phrase *that* was moved from some location inside the subtree of its sister node *S*.

$${}_{2}\,viewers\,{}_{3}\,will\,{}_{4}\,tune\,{}_{5}\,in\,{}_{6}\,to\,{}_{7}\,see\,{}_{8}$$

[1] Modeling conjunction requires an alternative definition for patterns: $ptrn : \mathcal{T} \times \mathcal{G} \rightarrow Powerset(\mathbf{\Gamma}) \cup \{none\}$. For the sake of simplicity, we ignore conjunctions in the following discussion, except for in the few places where it matters, since this has little impact on the development of our model.

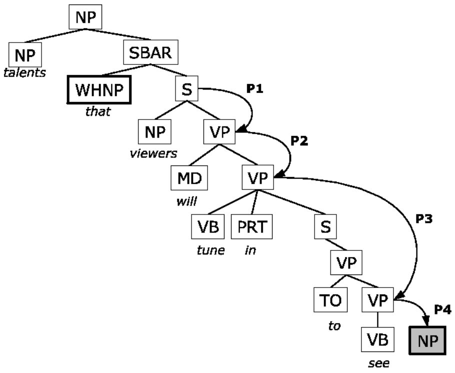


Figure 2: A pattern application example

Now suppose there is a pattern P_1 that matches the tree T_{28} indicating that the gap is somewhere in its subtree T_{38} (*will tune in to see*), i.e., $app(P_1, T_{28}) \rightarrow g^{38}_{38}$. The process of applying patterns continues until the pattern P_4 proposes an exact location for the gap: $app(P_4, T_{78}) = g^{78}_{88}$.

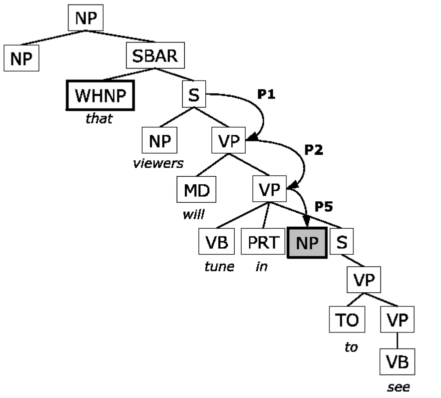


Figure 3: Another pattern application example

Suppose that, in addition to the pattern applications shown in Figure 2, there is one more, namely: $app(P_5, T_{48}) \rightarrow g^{48}_{66}$. The sequence of patterns P_1, P_2, P_5 proposes an alternative grammatically plausible location for the gap, as shown in Figure 3. Notice that the combination of the two sequences produces a tree of patterns, as shown in Figure 4, and this pattern tree covers much of the structure of the T_{28} subtree.

2.1 Tree Classes

The number of unique subtrees that contain WH-phrases is essentially infinite; hence, modeling them directly is infeasible. However, trees with varying details, e.g., optional adverbials, often can be characterized by the same tree of patterns. Hence, we can represent the space of trees by utilizing a relatively small set of classes of trees that are determined by their tree of pattern applications.

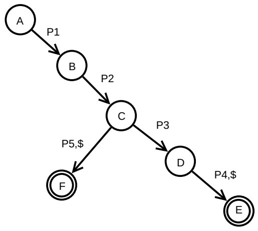


Figure 4: Pattern tree

Let Π be the set of all patterns. We define the set of patterns matching tree T_{ij} as follows:

$$M(T_{ij}) = \{P \mid P \in \Pi \wedge app(P, T_{ij}) \neq none\}$$

To enable recursive application:

$$app(ptrn, g^{ab}_{xy}) = \begin{cases} app(ptrn, T_{ab}) & \text{if } x < y \\ none & \text{if } x = y \end{cases}$$

A *Pattern Chain PC* is a sequence of pairs of patterns and sets of pattern sets, terminated by \$, i.e., $(\frac{p_1}{M_1}, \frac{p_2}{M_2}, ...\frac{p_n}{M_n}, \$)$, where $\forall_i \; p_i \in M_i \subset \Pi$. $M_i = M(T_{ab})$, where T_{ab} is the result of consequent application of the first $i-1$ patterns: $app(p_{i-1}, app(p_{i-2}, ..., app(p_1, T_{\alpha\beta}))) = g^{ab}_{xy}$, and where $T_{\alpha\beta}$ is the subtree we started with, (T_{28} in the example above). We define *the application of a pattern chain PC* $= (\frac{p_1}{M_1}, \frac{p_2}{M_2}, ...\frac{p_n}{M_n}, \$)$ to a tree T_{ij} as:

$$app(PC, T_{ij}) = app(p_n, ...app(p_2, app(p_1, T_{ij})))$$

It is important to also define a function to map a tree to the set of pattern chains applicable to a particular tree. The pseudocode for this function called FindPCs appears in Figure 5[2]. When applied to T_{ij}, this function returns the set of all pattern chains, applications of which would result in concrete gap locations. The algorithm is guaranteed to terminate as long as trees are of finite depth and each pattern moves the gap location down at least one level in the tree at each iteration. Using this function, we define *Tree Class* (TC) of a tree T_{ij} as $TC(T_{ij})$ = FindPCs(T_{ij}).

[2] $list \circ element$ means "append $element$ to $list$".

```
function FindPCs'($T_{ij}$, $PC$, $allPCs$) {
    $M_{ij} \leftarrow \{P \mid P \in \Pi \wedge app(P, T_{ij}) \neq none\}$
    forall $P \in M_{ij}$
        $g^{ab}_{xy} \leftarrow app(P, T_{ij})$
        $PC \leftarrow PC \circ \frac{P}{M_{ij}}$
        if $x = y$ then // $g^{ab}_{xy}$ is a concrete location
            $allPCs \leftarrow allPCs \cup \{PC \circ \$\}$
        else
            $allPCs \leftarrow$ FindPCs'($T_{ab}$, $PC$, $allPCs$)
    return $allPCs$ }
function FindPCs($T_{ij}$) { return FindPCs'($T_{ij}$, [ ], $\emptyset$) }
```

Figure 5: Pseudocode for FindPCs

In the case of a conjunction, the function FindPCs is slightly more complex. Recall that in this case $app(P, T_{ij})$ produces a set of gaps or *none*. The pseudocode for this case appears in Figure 6.

```
function CrossProd($PC_1$, $PC_2$) {
    $prod \leftarrow \emptyset$
    forall $pc_i \in PC_1$
        forall $pc_j \in PC_2$ : $prod \leftarrow prod \cup \{pc_i \circ pc_j\}$
    return $prod$ }
function FindPCs($T_{ij}$) {
    $M_{ij} \leftarrow \{P \mid P \in \Pi \wedge app(P, T_{ij}) \neq none\}$
    $newPCs \leftarrow \emptyset$
    forall $P \in M_{ij}$
        $PCs \leftarrow \{[\,]\}$
        forall $g^{ab}_{xy} \in app(P, T_{ij})$
            if $x = y$ then
                forall $pc \in PCs$ : $pc \leftarrow pc \circ \$$
            else
                $PCs \leftarrow$ CrossProd($PCs$,FindPCs($T_{ab}$))
        forall $pc \in PCs$ : $pc \leftarrow \frac{P}{M_{ij}} \circ pc$
        $newPCs \leftarrow newPCs \cup PCs$
    return $newPCs$ }
```

The set $app(P, T_{ij})$ must be ordered, so that branches of conjunction are concatenated in a well defined order.

Figure 6: Pseudocode for FindPCs in the case of conjunction

2.2 A Gap Automaton

The set of pattern chains constructed by the function FindPCs can be represented as a *pattern tree* with patterns being the edges. For example, the pattern tree in Figure 4 corresponds to the tree displayed in Figures 2 and 3.

This pattern tree captures the history of gap propagations beginning at A. Assuming at that point only pattern P_1 is applicable, subtree B is produced. If P_2 yields subtree C, and at that point patterns P_3 and P_5 can be applied, this yields subtree D and exact location F (which is expressed by the termination symbol $), respectively. Finally, pattern P_4 matches subtree D and proposes exact gap location E. It is important to note that this pattern tree can be thought of as an automaton, with $A, B, C, D, E,$ and F being the states and the pattern applications being the transitions.

Now, let us assign meaning of the states $A, B, C,$ and D to be the set of matching patterns, i.e., $A = \{P_1\}$, $B = \{P2\}$, $C = \{P_3, P_5\}$, $D = \{P_4\}$, and $E = F = \emptyset$. Given this representation, the pattern chains for the insertion of the gaps in our example would be as follows:

$$(\{P_1\}) \xrightarrow{P_1} (\{P_2\}) \xrightarrow{P_2} (\{P_3, P_5\}) \xrightarrow{P_3} (\{P_4\}) \xrightarrow{P_4,\$} (\emptyset)$$

$$(\{P_1\}) \xrightarrow{P_1} (\{P_2\}) \xrightarrow{P_2} (\{P_3, P_5\}) \xrightarrow{P_5,\$} (\emptyset)$$

With this representation, we can create a regular grammar using patterns as the terminals and their powerset as the non-terminals (adding a few more details like the start symbol) and production rules such as $\{P_2\} \rightarrow P_2\ \{P_3, P_5\}$. However, for our example the chain of patterns applied $P_1, P_2, P_3, P_4,\$$ could generate a pattern tree that is incompatible with the original tree. For example:

$$(\{P_1\}) \xrightarrow{P_1} (\{P_2\}) \xrightarrow{P_2} (\{P_3, P_5\}) \xrightarrow{P_3} (\{P_3, P_4\}) \xrightarrow{P_4,\$} (\emptyset)$$

which might correspond to something like "*that viewers will tune in to expect to see.*" Note that this pattern chain belongs to a different *tree class*, which incidentally would have inserted the gap at a different location (VP see gap).

To overcome this problem we add additional constraints to the grammar to ensure that all parses the grammar generates belong to the same tree class. One way to do this is to include the start state of a transition as an element of the terminal, e.g., $\frac{P_2}{\{P_2\}}$, $\frac{P_3}{\{P_3, P_5\}}$. That is, we extend the terminals to include the left-hand side of the productions they are emitted from, e.g.,

$$\{P_2\} \quad \rightarrow \quad \frac{P_2}{\{P_2\}}\ \{P_3, P_5\}$$

$$\{P_3, P_5\} \quad \rightarrow \quad \frac{P_3}{\{P_3, P_5\}}\ \{P_4\}$$

and the sequence of terminals becomes: $\frac{P_1}{\{P_1\}}\ \frac{P_2}{\{P_2\}}\ \frac{P_3}{\{P_3,P_5\}}\ \frac{P_4}{\{P_4\}}$ \$.

Note that the grammar is unambiguous. For such a grammar, the question "what is the probability of a parse tree given a string and grammar" doesn't make sense; however, the question "what is the probability of a string given the grammar" is still valid, and this is essentially what we require to develop a generative model for gap insertion.

2.3 The Pattern Grammar

Let us define the pattern grammar more rigorously. Let Π be the set of patterns, and $\tilde{\Pi} \subset \Pi$ be the set of *terminal* patterns[3]. Let $pset(P)$ be the set of all subsets of patterns which include the pattern P, i.e., $pset(P) = \{\nu \cup \{P\} \mid \nu \in powerset(\Pi)\}$

- Let $T = \{\frac{P}{pset(P)} \mid P \in \Pi\} \bigcup \{\$\}$ be the set of terminals, where \$ is a special symbol[4].
- Let $N = \{S\} \bigcup powerset(\Pi)$ be the set of nonterminals with S being the start symbol.
- Let P be the set of productions, defined as the union of the following sets:
  1. $\{S \rightarrow \nu \mid \nu \in powerset(\Pi)\}$.
  2. $\{\nu \rightarrow \frac{P}{\nu}\ \mu \mid P \in \Pi - \tilde{\Pi}\ ,\ \nu \in pset(P)$ and $\mu \in powerset(\Pi)\}$. These are nonterminal transitions, note that they emit only non-terminal patterns.
  3. $\{\nu \rightarrow \frac{P}{\nu}\$ \mid P \in \tilde{\Pi}$ and $\nu \in pset(P)\}$. These are the terminal transitions, they emit a terminal pattern and the symbol \$.
  4. $\{\nu \rightarrow \frac{P}{\nu}\ \mu_1 \ldots \mu_n \mid P \in \Pi - \tilde{\Pi}\ ,\ \nu \in pset(P)$ and $\forall_{i \in [1..n]}\ \mu_i \in powerset(\Pi)\}$. This rule models conjunction with n branches.

2.4 Our Gap Model

Given the grammar defined in the previous subsection, we will define a probabilistic model for gap insertion. Recall that our goal is to find:

$$\underset{x,a,b}{argmax}\ Pr(g^{ab}_{xx}|T)$$

Just like the probability of a sentence is obtained by summing up the probabilities of its parses, the probability of the gap being at g^{ab}_{xx} is the sum of probabilities of all pattern chains that yield g^{ab}_{xx}.

$$Pr(g^{ab}_{xx}|T) = \sum_{pc_i \in \Upsilon} Pr(pc_i|T)$$

where $\Upsilon = \{pc \mid app(pc, T) = g^{ab}_{xx}\}$. Note that $pc_i \in TC(T)$ by definition.

For our model, we use two approximations. First, we collapse a tree T into its Tree Class $TC(T)$, effectively ignoring details irrelevant to gap insertion:

$$Pr(pc_i|T) \approx Pr(pc_i|TC(T))$$

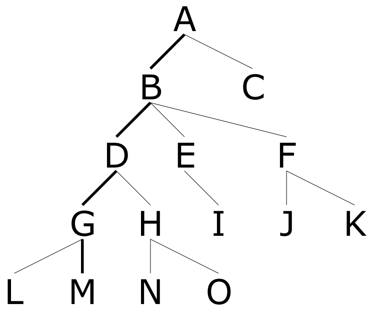


Figure 7: A pattern tree with the pattern chain $ABDGM$ marked using bold lines

Consider the pattern tree shown in Figure 7. The probability of the pattern chain $ABDGM$ given the pattern tree can be computed as:

$$\begin{aligned} Pr(ABDGM|TC(T)) &= \frac{Pr(ABDGM, TC(T))}{Pr(TC(T))} \\ &= \frac{\text{NR}(ABDGM, TC(T))}{\text{NR}(TC(T))} \end{aligned}$$

where NR$(TC(T))$ is the number of occurrences of the tree class $TC(T)$ in the training corpus and NR$(ABDGM, TC(T))$ is the number cases when the pattern chain $ABDGM$ leads to a correct gap in trees corresponding to the tree class $TC(T)$. For many tree classes, NR$(TC(T))$ may be a small number or even zero, thus this direct approach cannot be applied to the estimation of $Pr(pc_i|TC(T))$. Further approximation is required to tackle the sparsity issue.

In the following discussion, XY will denote an edge (pattern) between vertices X and Y in

[3] Patterns that generate exact position for a gap.

[4] Symbol \$ helps to separate branches in strings with conjunction.

the pattern tree shown in Figure 7. Note that $Pr(ABDGM|TC(T))$ can be represented as:

$$Pr(AB|TC(T),A) \times Pr(BD|TC(T),AB) \times \\ \times Pr(DG|TC(T),ABD) \times Pr(GM|TC(T),ABDG)$$

We make an independence assumption, specifically, that $Pr(BD|TC(T),AB)$ depends only on states B, D, and the edge between them, not on the whole pattern tree or the edges above B, i.e., $Pr(BD|TC(T),AB) \approx Pr(BD,D|B)$. Note that this probability is equivalent to the probability of a production $Pr(B \overset{BD}{\rightarrow} D)$ of a PCFG.

Recall that the meaning assigned to a state in pattern grammar in Section 2.2 is the set of patterns matching at that state. Thus, according to that semantics, only the edges displayed bold in Figure 8 are involved in computation of $Pr(B \overset{BD}{\rightarrow} D)$. Written in the style we used for our grammar, the production is $\{BD,BE,BF\} \rightarrow \frac{BD}{\{BD,BE,BF\}}\{DG,DH\}$.

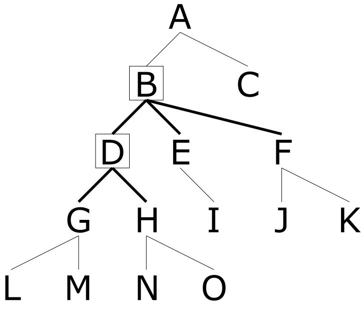


Figure 8: The context considered for estimation of the probability of transition from B to D

Pattern trees are fairly shallow (partly because many patterns cover several layers in a parse tree as can be seen in Figures 1 and 2); therefore, the context associated with a production covers a good part of a pattern tree. Another important observation is that the local configuration of a node, which is described by the set of matching patterns, is the most relevant to the decision of where the gap is to be propagated[5]. This is the reason why the states are represented this way.

Formally, the second approximation we make is as follows:

$$Pr(pc_i|TC(T)) \approx Pr(pc_i|G)$$

where G is a PCFG model based on the grammar described above.

$$Pr(pc_i|G) = \prod_{prod_j \in \mathcal{P}(pc_i)} Pr(prod_j|G)$$

where $\mathcal{P}(pc_i)$ is *the* parse of the pattern chain pc_i which is a string of terminals of G. Combining the formulae:

$$Pr(g^{ab}_{xx}|T) \approx \sum_{pc_i \in \Upsilon} Pr(pc_i|G)$$

Finally, since $Pr(TC(T)|G)$ is a constant for T,

$$\underset{x,a,b}{argmax}\ Pr(g^{ab}_{xx}|T) \approx \underset{x,a,b}{argmax} \sum_{pc_i \in \Upsilon} Pr(pc_i|G)$$

To handle conjunction, we must express the fact that pattern chains yield sets of gaps. Thus, the goal becomes:

$$\underset{(x_1,a_1,b_1),\dots,(x_n,a_n,b_n)}{argmax} Pr(\{g^{a_1b_1}_{x_1x_1},\dots,g^{a_nb_n}_{x_nx_n}\}|T)$$

$$Pr(\{g^{a_1b_1}_{x_1x_1},\dots,g^{a_nb_n}_{x_nx_n}\}|T) = \sum_{pc_i \in \Upsilon} Pr(pc_i|T)$$

where $\Upsilon = \{pc \mid app(pc,T) = \{g^{a_1b_1}_{x_1x_1},\dots,g^{a_nb_n}_{x_nx_n}\}\}$. The remaining equations are unaffected.

2.5 Smoothing

Even for the relatively small number of patterns, the number of non-terminals in the grammar can potentially be large ($2^{|\Pi|}$). This does not happen in practice since most patterns are mutually exclusive. Nonetheless, productions, unseen in the training data, do occur and their probabilities have to be estimated. Rewriting the probability of a transition $Pr(A \rightarrow \frac{a}{A} B)$ as $\mathcal{P}(A,a,B)$, we use the following interpolation:

$$\begin{aligned}\tilde{\mathcal{P}}(A,a,B) &= \lambda_1\mathcal{P}(A,a,B) + \lambda_2\mathcal{P}(A,a) \\ &+\lambda_3\mathcal{P}(A,B) + \lambda_4\mathcal{P}(a,B) + \lambda_5\mathcal{P}(a)\end{aligned}$$

We estimate the parameters on the held out data (section 24 of WSJ) using a hill-climbing algorithm.

[5] We have evaluated a model that only uses $Pr(BD|\{BD,BE,BF\})$ for the probability of taking BD and found it performs only slightly worse than the model presented here.

3 Evaluation

3.1 Setup

We compare our algorithm under a variety of conditions to the work of (Johnson, 2002) and (Gabbard et al., 2006). We selected these two approaches because of their availability[6]. In addition, (Gabbard et al., 2006) provides state-of-the-art results. Since we only model the insertion of WH-traces, all metrics include co-indexation with the correct WH phrases identified by their type and word span.

We evaluate on three metrics. The first metric, which was introduced by Johnson (2002), has been widely reported by researchers investigating gap insertion. A gap is scored as correct only when it has the correct type and string position. The metric has the shortcoming that it does not require correct attachment into the tree.

The second metric, which was developed by Campbell (2004), scores a gap as correct only when it has the correct gap type and its mother node has the correct nonterminal label and word span. As Campbell points out, this metric does not restrict the position of the gap among its siblings, which in most cases is desirable; however, in some cases (e.g., double object constructions), it does not correctly detect errors in object order. This metric is also adversely affected by incorrect attachments of optional constituents, such as PPs, due to the span requirement.

To overcome the latter issue with Campbell's metric, we propose to use a third metric that evaluates gaps with respect to correctness of their lexical head, type of the mother node, and the type of the co-indexed wh-phrase. This metric differs from that used by Levy and Manning (2004) in that it counts only the dependencies involving gaps, and so it represents performance of the gap insertion algorithm more directly.

We evaluate gap insertion on gold trees from section 23 of the Wall Street Journal Penn Treebank (WSJ) and parse trees automatically produced using the Charniak (2000) and Bikel (2004) parsers. These parsers were trained using sections 00 through 22 of the WSJ with section 24 as the development set.

Because our algorithm inserts only traces of non-empty WH phrases, to fairly compare to Johnson's and Gabbard's performance on WH-traces alone, we remove the other gap types from both the gold trees and the output of their algorithms. Note that Gabbard et al.'s algorithm requires the use of function tags, which are produced using a modified version of the Bikel parser (Gabbard et al., 2006) and a separate software tool (Blaheta, 2003) for the Charniak parser output.

For our algorithm, we do not utilize function tags, but we automatically replace the tags of auxiliary verbs in tensed constructions with *AUX* prior to inserting gaps using tree surgeon (Levy and Andrew, 2006). We found that Johnson's algorithm more accurately inserts gaps when operating on auxified trees, and so we evaluate his algorithm using these modified trees.

In order to assess robustness of our algorithm, we evaluate it on a corpus of a different genre – Broadcast News Penn Treebank (BN), and compare the result with Johnson's and Gabbard's algorithms. The BN corpus uses a modified version of annotation guidelines, with some of the modifications affecting gap placement.

```
Treebank 2 guidelines (WSJ style):
(SBAR (WHNP-2 (WP whom))
  (S (NP-SBJ (PRP they))
    (VP (VBD called)
      (S (NP-SBJ (-NONE- *T*-2))
         (NP-PRD (NNS exploiters))))))
```

```
Treebank 2a guidelines (BN style):
(SBAR-NOM (WHNP-1 (WP what))
 (S (NP-SBJ (PRP they))
  (VP (VBP call)
    (NP-2 (-NONE- *T*-1))
      (S-CLR (NP-SBJ (-NONE- *PRO*-2))
        (NP-PRD (DT an) (NN epidemic))))))
```

Since our algorithms were trained on WSJ, we apply tree transformations to the BN corpus to convert these trees to WSJ style. We also auxify the trees as described previously.

3.2 Results

Table 1 presents gap insertion F measure for Johnson's (2002) (denoted J), Gabbard's (2006) (denoted G), and our (denoted Pres) algorithms on section 23 gold trees, as well as on parses generated by the Charniak and Bikel parsers. In addition to WHNP and WHADVP results that are reported in the literature, we also present results for WHPP gaps even though there is a small number of them in section 23 (i.e., 22 gaps total). Since there are only 3 non-empty WHADJP phrases in section 23, we omit them in our evaluation.

[6] Johnson's source code is publicly available, and Ryan Gabbard kindly provided us with output trees produced by his system.

		Gold Trees			Charniak Parser			Bikel Parser		
	Metric	J	G	Pres	J	G	Pres	J	G	Pres
WHNP	Johnson	94.8	90.7	97.9	89.8	86.3	91.5	90.2	86.8	92.6
	Campbell	94.8	97.0	99.1	81.9	83.8	83.5	80.7	81.5	82.2
	Head dep	94.8	97.0	99.1	88.8	90.6	91.0	89.1	91.4	92.3
WHADVP	Johnson	75.5	91.4	96.5	61.4	78.0	80.0	61.0	77.9	77.2
	Campbell	74.5	89.1	95.0	61.4	71.7	78.4	60.0	71.5	74.8
	Head dep	75.5	89.8	95.8	64.4	78.0	84.7	63.0	77.1	80.3
WHPP	Johnson	58.1	N/R	72.7	35.7	N/R	55.0	42.9	N/R	53.7
	Campbell	51.6	N/R	86.4	28.6	N/R	60.0	35.7	N/R	63.4
	Head dep	51.6	N/R	86.4	35.7	N/R	70.0	35.7	N/R	73.2

Table 1: F1 performance on section 23 of WSJ (N/R indicates not reported)

Compared to Johnson's and Gabbard's algorithm, our algorithm significantly reduces the error on gold trees (table 1). Operating on automatically parsed trees, our system compares favorably on all WH traces, using all metrics, except for two instances: Gabbard's algorithm has better performance on WHNP, using Cambpell's metric and trees generated by the Charniak parser by 0.3% and on WHADVP, using Johnson's metric and trees produces by the Bikel parser by 0.7%. However, we believe that the dependency metric is more appropriate for evaluation on automatically parsed trees because it enforces the most important aspects of tree structure for evaluating gap insertion. The relatively poor performance of Johnson's and our algorithms on WHPP gaps compared that on WHADVP gaps is probably due, at least in part, to the significantly smaller number of WHPP gaps in the training corpus and the relatively wider range of possible attachment sites for the prepositional phrases.

Table 2 displays how well the algorithms trained on WSJ perform on BN. A large number of the errors are due to *FRAG*s which are far more common in the speech corpus than in WSJ. WHPP and WHADJP, although more rare than the other types, are presented for reference.

3.3 Error Analysis

It is clear from the contrast between the results based on gold standard trees and the automatically produced parses in Table 1 that parse error is a major source of error. Parse error impacts all of the metrics, but the patterns of errors are different. For WHNPs, Campbell's metric is lower than the other two across all three algorithms, suggesting that this metric is adversely affected by factors that do not impact the other metrics (most likely the span of the gap's mother node). For WHADVPs, the metrics show a similar degradation due to parse error across the board. We are reluctant to draw conclusions for the metrics on WHPPs; however, it should be noted that the position of the PP should be less critical for evaluating these gaps than their correct attachment, suggesting that the head dependency metric would more accurately reflect the performance of the system for these gaps.

Campbell's metric has an interesting property: in parse trees, we can compute the upper bound on recall by simply checking whether the correct WH-phrase and gap's mother node exist in the parse tree. We present recall results and upper bounds in Table 3. Clearly the algorithms are performing close to the upper bound for WHNPs when we take into account the impact of parse errors on this metric. Clearly there is room for improvement for the WHPPs.

	Metric	J	G	Pres
WHNP	Johnson	88.0	90.3	92.0
	Campbell	88.2	94.0	95.3
	Head dep	88.3	94.0	95.3
WHADVP	Johnson	76.4	92.0	94.3
	Campbell	76.3	88.2	92.4
	Head dep	76.3	88.5	92.5
WHPP	Johnson	56.6	N/R	75.7
	Campbell	60.4	N/R	91.9
	Head dep	60.4	N/R	91.9
WHADJP	Johnson	N/R	N/R	89.8
	Campbell	N/R	N/R	85.7
	Head dep	N/R	N/R	85.7

Table 2: F1 performance on gold trees of BN

In addition to parser errors, which naturally have the most profound impact on the performance, we found the following sources of errors to have impact on our results:

- Annotation errors and inconsistency in PTB, which impact not only the training of our system, but also its evaluation.

Charniak Parser	J	G	Pres	UB
WHNP	81.9	82.8	83.5	84.0
WHADVP	61.4	71.7	78.4	81.1
WHPP	28.6	N/R	60.0	86.4

Bikel Parser	J	G	Pres	UB
WHNP	77.0	80.5	81.5	82.0
WHADVP	47.2	70.1	74.8	78.0
WHPP	22.7	N/R	59.1	81.8

Table 3: Recall on trees produced by the Charniak and Bikel parsers and their upper bounds (UB)

1. There are some POS labeling errors that confuse our patterns, e.g.,
```
(SBAR (WHNP-3 (IN that))
    (S (NP-SBJ (NNP Canada))
        (VP (NNS exports)
            (NP (-NONE- *T*-3))
            (PP ...))))
```

2. Some WHADVPs have gaps attached in the wrong places or do not have gaps at all, e.g.,
```
(SBAR (WHADVP (WRB when))
    (S (NP (PRP he))
        (VP (VBD arrived)
            (PP (IN at)
                (NP ...))
            (ADVP (NP (CD two)
                    (NNS days))
                (JJ later)))))
```

3. PTB annotation guidelines leave it to annotators to decide whether the gap should be attached at the conjunction level or inside its branches (Bies et al., 1995) leading to inconsistency in attachment decisions for adverbial gaps.

- Lack of coverage: Even though the patterns we use are very expressive, due to their small number some rare cases are left uncovered.

- Model errors: Sometimes despite one of the applicable pattern chains proposes the correct gap, the probabilistic model chooses otherwise. We believe that a lexicalized model can eliminate most of these errors.

4 Conclusions and Future Work

The main contribution of this paper is the development of a generative probabilistic model for gap insertion that operates on subtree structures. Our model achieves state-of-the-art performance, demonstrating results very close to the upper bound on WHNP using Campbell's metric. Performance for WHADVPs and especially WHPPs, however, has room for improvement.

We believe that lexicalizing the model by adding information about lexical heads of the gaps may resolve some of the errors. For example:

```
(SBAR (WHADVP-3 (WRB when))
  (S (NP (NNP Congress))
    (VP (VBD wanted)
      (S (VP (TO to)
        (VP (VB know) ...)))
      (ADVP (-NONE- *T*-3)))))

(SBAR (WHADVP-1 (WRB when))
 (S (NP (PRP it))
   (VP (AUX is)
     (VP (VBN expected)
       (S (VP (TO to)
            (VP (VB deliver) ...
              (ADVP (-NONE- *T*-1)))))))))
```

These sentences have very similar structure, with two potential places to insert gaps (ignoring reordering with siblings). The current model inserts the gaps as follows: *when Congress (VP wanted (S to know) gap)* and *when it is (VP expected (S to deliver) gap)*, making an error in the second case (partly due to the bias towards shorter pattern chains, typical for a PCFG). However, *deliver* is more likely to take a temporal modifier than *know*.

In future work, we will investigate methods for adding lexical information to our model in order to improve the performance on WHADVPs and WHPPs. In addition, we will investigate methods for automatically inferring patterns from a treebank corpus to support fast porting of our approach to other languages with treebanks.

5 Acknowledgements

We would like to thank Ryan Gabbard for providing us output from his algorithm for evaluation. We would also like to thank the anonymous reviewers for invaluable comments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

A. Bies, M. Ferguson, K. Katz, and R. MacIntyre. 1995. Bracketing guidelines for treebank II style Penn Treebank project. Technical report.

D. M. Bikel. 2004. *On the Parameter Space of Gen-*

erative Lexicalized Statistical Parsing Models. Ph.D. thesis, University of Pennsylvania.

D. Blaheta. 2003. *Function Tagging*. Ph.D. thesis, Brown University.

R. Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

M. Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

P. Dienes and A. Dubey. 2003. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*.

R. Gabbard, S. Kulick, and M. Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

D. Higgins. 2003. A machine-learning approach to the identification of WH gaps. In *Proceedings of the Annual Meeting of the European Chapter of the Association for Computational Linguistics*.

M. Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

R. Levy and G Andrew. 2006. Tregex and Tsurgeon: Tools for querying and manipulating tree data structures. In *Proceedings of LREC*.

R. Levy and C. Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

W. Wang and M. P. Harper. 2002. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources in language modeling. In *Proceedings of the Empirical Methods in Natural Language Processing*.

W. Wang, M. P. Harper, and A. Stolcke. 2003. The robustness of an almost-parsing language model given errorful training data. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Treebank Annotation Schemes and Parser Evaluation for German

Ines Rehbein
NCLT
School of Computing, DCU,
Dublin, Ireland
irehbein@computing.dcu.ie

Josef van Genabith
NCLT,
School of Computing, DCU,
Dublin, Ireland
IBM Dublin Center for Advanced Studies
josef@computing.dcu.ie

Abstract

Recent studies focussed on the question whether less-configurational languages like German are harder to parse than English, or whether the lower parsing scores are an artefact of treebank encoding schemes and data structures, as claimed by Kübler et al. (2006). This claim is based on the assumption that PARSEVAL metrics fully reflect parse quality across treebank encoding schemes. In this paper we present new experiments to test this claim. We use the PARSEVAL metric, the Leaf-Ancestor metric as well as a dependency-based evaluation, and present novel approaches measuring the effect of controlled error insertion on treebank trees and parser output. We also provide extensive past-parsing cross-treebank conversion. The results of the experiments show that, contrary to Kübler et al. (2006), the question whether or not German is harder to parse than English remains undecided.

1 Introduction

A long-standing and unresolved issue in the parsing literature is whether parsing less-configurational languages is harder than e.g. parsing English. German is a case in point. Results from Dubey and Keller (2003) suggest that state-of-the-art parsing scores for German are generally lower than those obtained for English, while recent results from Kübler et al. (2006) raise the possibility that this might be an artefact of particular encoding schemes and data structures of treebanks, which serve as training resources for probabilistic parsers. Kübler (2005) and Maier (2006) show that treebank annotation schemes have considerable influence on parsing results. A comparison of unlexicalised PCFG parsing (Kübler, 2005) trained and evaluated on the German NEGRA (Skut et al., 1997) and the TüBa-D/Z (Telljohann et al., 2004) treebanks using LoPar (Schmid, 2000) shows a difference in parsing results of about 16%, using the PARSEVAL metric (Black et al., 1991). Kübler et al. (2006) conclude that, contrary to what had been assumed, German is not actually harder to parse than English, but that the NEGRA annotation scheme does not support optimal PCFG parsing performance.

Despite being the standard metric for measuring PCFG parser performance, PARSEVAL has been criticised for not representing 'real' parser quality (Carroll et al., 1998; Brisco et al., 2002; Sampson and Babarbczy, 2003). PARSEVAL checks label and wordspan identity in parser output compared to the original treebank trees. It neither weights results, differentiating between linguistically more or less severe errors, nor does it give credit to constituents where the syntactic categories have been recognised correctly but the phrase boundary is slightly wrong.

With this in mind, we question the assumption that the PARSEVAL results for NEGRA and TüBa-D/Z reflect a real difference in quality between the parser output for parsers trained on the two different treebanks. As a consequence we also question the conclusion that PARSEVAL results for German in the same range as the parsing results for the English

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 630–639, Prague, June 2007. ©2007 Association for Computational Linguistics

Penn-II Treebank prove that German is not harder to parse than the more configurational English. To investigate this issue we present experiments on the German TIGER treebank (Dipper et al., 2001) and the TüBa-D/Z treebank. TIGER is based on and extends the NEGRA data and annotation scheme. Our error insertion and past-parsing treebank-encoding experiments experiments show that the differences in parsing results for the two treebanks are not caused by a higher number of errors in the output of the parser trained on the TIGER treebank, but are due to the bias of the PARSEVAL metric towards annotation schemes (such as that of TüBa-D/Z) with a higher ratio of non-terminal/terminal nodes. The experiments also show that compared to PARSEVAL the Leaf-Ancestor metric is somewhat less susceptible to non-terminal/terminal ratios and that contrary to the PARSEVAL results, dependency-based evaluations score TIGER trained parsers higher than TüBa-D/Z trained parsers.

This paper is structured as follows: Section 2 gives an overview of the main features of the two treebanks. Section 3 describes our first experiment, where we systematically insert controlled errors into the original treebank trees and compare the influence of these modifications on the evaluation results in the PARSEVAL metric and the Leaf-Ancestor metric against the original, unmodified trees for both treebanks. In Section 4 we present the second experiment, where we extract an unlexicalised PCFG from each of the treebanks. Then we convert the output of the PCFG parser trained on the TüBa-D/Z into a TIGER-style format and evaluate the converted trees. In Section 5 we present a dependency-based evaluation and compare the results to the results of the two other measures. The last section concludes.

2 The TIGER Treebank and the TüBa-D/Z

The two German treebanks used in our experiments are the TIGER Treebank (Release 2) and the Tüba-D/Z (Release 2). The TüBa-D/Z consists of approximately 22 000 sentences, while the TIGER Treebank is much larger with more than 50 000 sentences. Both treebanks contain German newspaper text and are annotated with phrase structure and dependency (functional) information. Both treebanks use the Stuttgart Tübingen POS Tag Set (Schiller et al., 95). TIGER uses 49 different grammatical function labels, while the TüBa-D/Z utilises only 36 function labels. For the encoding of phrasal node categories the TüBa-D/Z uses 30 different categories, the TIGER Treebank uses a set of 27 category labels.

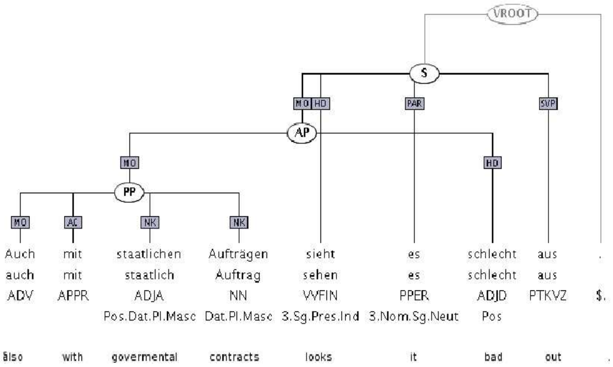
Other major differences between the two treebanks are: in the Tiger Treebank long distance dependencies are expressed through crossing branches (Figure 1), while in the TüBa-D/Z the same phenomenon is expressed with the help of grammatical function labels (Figure 2), where the node label V-MOD encodes the information that the PP modifies the verb. The annotation in the Tiger Treebank is rather flat and allows no unary branching, whereas the nodes in the TüBa-D/Z do contain unary branches and a more hierarchical structure, resulting in a much deeper tree structure than the trees in the Tiger Treebank. This results in an average higher number of nodes per sentence for the TüBa-D/Z. Table 1 shows the differences in the ratio of nodes for the Tiger treebank and the TüBa-D/Z.

	phrasal nodes/sent	**phrasal nodes/word**	**words /sent**
TIGER	8.29	0.47	17.60
TüBa-D/Z	20.69	1.20	17.27

Table 1: Average number of phrasal nodes/words in TIGER and TüBa-D/Z

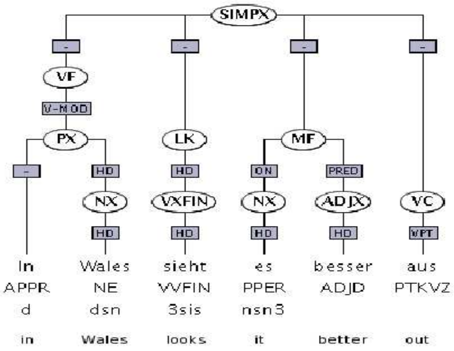
Figures 1 and 2 also illustrate the different annotation of PPs in both annotation schemes. In the Tiger treebank the internal structure of the PP is flat and the adjective and noun inside the PP are directly attached to the PP, while the TüBa-D/Z is more hierarchical and inserts an additional NP node.

Another major difference is the annotation of topological fields in the style of Drach (1937) and Höhle (1986) in the TüBa-D/Z. The model captures German word order, which accepts three possible sentence configurations (verb first, verb second and verb last), by providing fields like the initial field (VF), the middle field (MF) and the final field (NF). The fields are positioned relative to the verb, which can fill in the left (LK) or the right sentence bracket (VC). The ordering of topological fields is determined by syntactic constraints.


VROOT
S
MO
HD
PAR
SVP
AP
MO
HD
PP
MO
AC
NK
NK
Auch mit staatlichen Aufträgen sieht es schlecht aus .
auch mit staatlich Auftrag sehen es schlecht aus
ADV APPR ADJA NN VVFIN PPER ADJD PTKVZ $.
Pos.Dat.Pl.Masc Dat.Pl.Masc 3.Sg.Pres.Ind 3.Nom.Sg.Neut Pos
also with govermental contracts looks it bad out .

Auch mit staatlichen Aufträgen sieht es schlecht aus.
"It also looks bad for public contracts."

Figure 1: TIGER treebank tree


SIMPX
-
-
-
-
VF
V-MOD
PX
LK
MF
-
HD
HD
ON
PRED
NX
VXFIN
NX
ADJX
VC
HD
HD
HD
HD
VPT
In Wales sieht es besser aus
APPR NE VVFIN PPER ADJD PTKVZ
d dsn 3sis nsn3
in Wales looks it better out

In Wales sieht es besser aus.
"Things seem better in Wales."

Figure 2: TüBa-D/Z treebank tree

2.1 Differences between TIGER and NEGRA

To date, most PCFG parsing for German has been done using the NEGRA corpus as a training resource. The flat annotation scheme of the TIGER treebank is based on the NEGRA annotation scheme, but it also employs some important extensions, which include the annotation of verb-subcategorisation, appositions and parentheses, coordinations and the encoding of proper nouns (Brants et al., 2002).

3 Treebank Preprocessing: Converting TIGER Graphs into CFG Trees

The sentences in the TIGER treebank are represented as graphs with LDDs expressed through crossing branches. Before being able to insert errors or extract a PCFG we had to resolve these crossing branches in the TIGER treebank. This was done by attaching the non-head child nodes higher up in the tree, following Kübler (2006). For the graph in Figure 1 this would mean that the modifying PP *"Auch mit staatlichen Aufträgen"* (also for public contracts) was attached directly to the S node, while the head of the adjectival phrase (AP) remained in it's original position. As a side effect this leads to the creation of some unary nodes in the TIGER trees. We also inserted a virtual root node and removed all functional labels from the TIGER and TüBa-D/Z trees.

4 Experiment I

Experiment I is designed to assess the impact of identical errors on the two treebank encoding schemes and the PARSEVAL[1] and Leaf-Ancestor evaluation metrics.

4.1 Experimental Setup

The TIGER treebank and the TüBa-D/Z both contain newspaper text, but from different German newspapers. To support a meaningful comparison we have to compare similar sentences from both treebanks. In order to control for similarity we selected all sentences of length $10 \leq n \leq 40$ from both treebanks. For all sentences with equal length we computed the average number of prepositions, determiners, nouns (and related POS such as proper names and personal pronouns), interrogative pronouns, finite verbs, infinite verbs, past participles and imperative verb forms. For each sentence length we selected all sentences from both treebanks which showed an average for each of the POS listed above which did not deviate more than 0.8 from the average for all sentences for this particular sentence length. From this set we randomly selected 1024 sentences for each of the treebanks. This results in two test sets, comparable in word length, syntactic structure and complexity. Table 2 shows the ratio of phrasal versus terminal nodes in the test sets.

We then inserted different types of controlled errors automatically into the *original* treebank trees in our test sets and evaluated the modified trees against

[1] In all our experiments we use the `evalb` metric (Sekine and Collins, 1997), the most commonly used implementation of the PARSEVAL metric.

	phrasal nodes/sent	phrasal nodes nodes/word	words /sent
TIGER	6.97	0.48	14.49
TüBa-D/Z	19.18	1.30	14.75

Table 2: Average number of phrasal nodes/words in the TIGER and TüBa-D/Z test set

the original treebank trees, in order to assess the impact of similar (controlled for type and number) errors on the two encoding schemes.

4.2 Error Insertion

The errors fall into three types: attachment, span and labeling (Table 3). We carried out the same number of error insertions in both test sets.

	Error description
ATTACH I	Attach PPs inside an NP one level higher up in the tree
ATTACH II	Change verb attachment to noun attachment for PPs on sentence level, inside a VP or in the MF (middle field)
LABEL I	Change labels of PPs to NP
LABEL II	Change labels of VPs to PP
SPAN I	Include adverb to the left of a PP into the PP
SPAN II	Include NN to the left of a PP into the PP
SPAN III	Combination of SPANI and SPANII

Table 3: Description of inserted error types

4.3 Results for Error Insertion for the Original Treebank Trees

Table 4 shows the impact of the error insertion into the original treebank trees on PARSEVAL results, evaluated against the gold trees. PARSEVAL results in all experiments report labelled precision and recall. The first error (PP attachment I, 85 insertions in each test set) leads to a decrease in f-score of 1.16 for the TIGER test set, while for the TüBa-D/Z test set the same error only caused a decrease of 0.43. The effect remains the same for all error types and is most pronounced for the category label errors, because the frequency of the labels resulted in a large number of substitutions. The last row lists the total weighted average for all error types, weighted with respect to their frequency of occurrence in the test sets.

Table 4 clearly shows that the PARSEVAL measure punishes the TIGER treebank annotation scheme to a greater extent, while the same number and type of errors in the TüBa-D/Z annotation scheme does not have an equally strong effect on PARSEVAL results for similar sentences.

	TIGER	TüBa	# errors
PP attachment I	98.84	99.57	85
PP attachment II	98.75	99.55	89
Label I	80.02	92.73	1427
Label II	93.00	97.45	500
SPAN I	99.01	99.64	71
SPAN II	97.47	99.08	181
SPAN III	96.51	98.73	252
total weighted ave.	87.09	95.30	

Table 4: f-score for PARSEVAL results for error insertion in the original treebank trees

4.4 Discussion: PARSEVAL and LA

Experiment I shows that the gap between the PARSEVAL results for the two annotation schemes does not reflect a difference in quality between the trees. Both test sets contain the same number of sentences with the same sentence length and are equivalent in complexity and structure. They contain the same number and type of errors. This suggests that the difference between the results for the TIGER and the TüBa-D/Z test set are due to the higher ratio of non-terminal/terminal nodes in the TüBa-D/Z trees (Table 1).

In order to obtain an alternative view on the quality of our annotation schemes we used the leaf-ancestor (LA) metric (Sampson and Babarbczy, 2003), a parser evaluation metric which measures the similarity of the path from each terminal node in the parse tree to the root node. The path consists of the sequence of node labels between the terminal node and the root node, and the similarity of two paths is calculated by using the Levenshtein distance (Levenshtein, 1966). Table 5 shows the results for the leaf-ancestor evaluation metric for our error insertion test sets. Here the weighted average results for the two test sets are much closer to each other (94.98 vs. 97.18 as against 87.09 vs. 95.30). Only the label errors, due to the large numbers, show a significant difference between the two annotation schemes. Tables 4 and 5 show that compared to PARSEVAL the LA metric is somewhat less sensitive to the nonterminal/terminal ratio.

Figure 3 illustrates the different behaviour of the

	TIGER	**TüBa**	**# errors**
PP attachment I	99.62	99.70	85
PP attachment II	99.66	99.78	89
Label I	92.45	95.24	1427
Label II	96.05	99.28	500
SPAN I	99.82	99.84	71
SPAN II	99.51	99.77	181
SPAN III	99.34	99.62	252
total weighted ave.	94.98	97.18	

Table 5: LA results for error insertion in the original treebank trees

two evaluation metrics with respect to an example sentence.

Sentence 9:
Die Stadtverwaltung von Venedig hat erstmals streunende Katzen gezählt.
"For the first time the city council of Venice has counted straying cats."

```
(TOP
    (S
      (NP
         (ART Die [the] )
         (NN Stadtverwaltung [city counsil] )
         (PP
            (APPR von [of] )
            (NE Venedig [Venice] )
         )
      )
      (VAFIN hat [has] )
      (VP
         (ADV erstmals [for the first time] )
         (NP
            (ADJA streunende [straying] )
            (NN Katzen [cats] )
         )
         (VVPP gezählt [counted] )
      )
    )
    ($. .)
)
```

Figure 3: Sentence 9 from the TIGER Test Set

Table 6 shows that all error types inserted into Sentence 9 in our test set result in the same evaluation score for the PARSEVAL metric, while the LA metric provides a more discriminative treatment of PP attachment errors, label errors and span errors for the same sentence (Table 6). However, the differences in the LA results are only indirectly caused by the different error types. They actually reflect the number of terminal nodes affected by the error insertion. For Label I and II the LA results vary considerably, because the substitution of the PP for an NP (Label I) in Figure 3 affects two terminal nodes only (PP *von* [of] *Venedig* [Venice]), while the change of the VP into a PP (Label II) alters the paths of four terminal nodes (VP *erstmals* [for the first time] *streunende* [straying] *Katzen* [cats] *gezählt* [counted]) and therefore has a much greater impact on the overall result for the sentence.

ERROR	PARSEVAL	LA
PP attachment I	83.33	96.30
Label I	83.33	96.00
Label II	83.33	91.00
SPAN II	83.33	96.40

Table 6: Evaluation results for Sentence 9

The TüBa-D/Z benefits from its overall higher ratio of nodes per sentence, resulting in a higher ratio of non-terminal/terminal nodes per phrase and the effect, that the inserted label error affects a smaller number of terminal nodes than in the TIGER test set for LA testing.

5 Experiment II

Kübler (2005) and Maier (2006) assess the impact of the different treebank annotation schemes on PCFG parsing by conducting a number of modifications converting the TüBa-D/Z into a format more similar to the NEGRA (and hence TIGER) treebank. After each modification they extract a PCFG from the modified treebank and measure the effect of the changes on parsing results. They show that with each modification transforming the TüBa-D/Z into a more NEGRA-like format the parsing results also become more similar to the results of the NEGRA treebank, i.e. the results get worse. Maier takes this as evidence that the TüBa-D/Z is more adequate for PCFG parsing. This assumption is based on the belief that PARSEVAL results fully reflect parse quality across different treebank encoding schemes. This is not always true, as shown in Experiment I.

In our second experiment we crucially change the order of events in the Kübler (2005), Maier (2006) and Kübler et al. (2006) experiments: We first extract an unlexicalised PCFG from each of the original treebanks. We then transform the output of the parser trained on the TüBa-D/Z into a format more similar to the TIGER Treebank. In contrast to Kübler (2005) and Maier (2006), who converted the treebank before extracting the grammars in order to measure the impact of single features like topological fields or unary nodes on PCFG parsing, we convert the trees in the parser *output* of a parser trained on the *original* unconverted treebank resources. *This allows us to preserve the basic syntactic structure and also the errors present in the output trees resulting from a potential bias in the original treebank training resources*. The results for the original parser output evaluated against the unmodified gold trees should not be crucially different from the results for the modified parser output evaluated against the modified gold trees.

5.1 Experimental Setup

For Experiment II we trained BitPar (Schmid, 2004), a parser for highly ambiguous PCFG grammars, on the two treebanks. The TüBa-D/Z training data consists of the 21067 treebank trees not included in the TüBa-D/Z test set. Because of the different size of the two treebanks we selected 21067 sentences from the TIGER treebank, starting from sentence 10000 (and excluding the sentences in the TIGER test set).

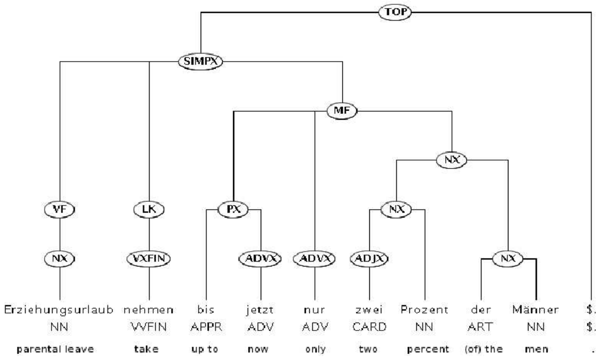
Before extracting the grammars we resolved the crossing branches in the TIGER treebank as described in Section 3. After this preprocessing step we extracted an unlexicalised PCFG from each of our training sets. Our TIGER grammar has a total of 21163 rule types, while the grammar extracted from the TüBa-D/Z treebank consists of 5021 rules only. We parsed the TIGER and TüBa-D/Z test set with the extracted grammars, using the gold POS tags for parser input. We then automatically converted the TüBa-D/Z output to a TIGER-like format and compare the evaluation results for the unmodified trees against the gold trees with the results for the converted parser output against the converted gold trees.

5.2 Converting the TüBa-D/Z Trees

The automatic conversion of the TüBa-D/Z-style trees includes the removal of topological fields and unary nodes as well as the deletion of NPs inside of PPs, because the NP child nodes are directly attached to the PP in the TIGER annotation scheme. As a last step in the conversion process we adapted the TüBa-D/Z node labels to the TIGER categories.

5.2.1 The Conversion Process: An Example

We demonstrate the conversion process using an example sentence from the TüBa-D/Z test set (Figure 4). The converted tree is given in Figure 5: topological fields, here VF (initial field), MF (middle field) and LK (left sentence bracket), as well as unary nodes have been removed. The category labels have been changed to TIGER-style annotation.



Erziehungsurlaub nehmen bisher nur zwei Prozent der Männer.
"Until now only two percent of the men take parental leave."

Figure 4: Original TüBa-D/Z-style gold tree

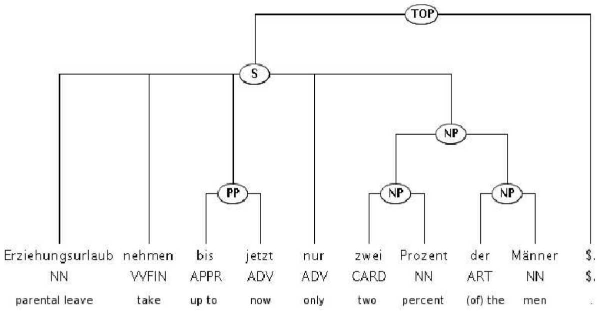


Figure 5: Converted TIGER-style gold tree

Figure 6 shows the unmodified parser output from the TüBa-D/Z trained grammar for the same string. The parser incorrectly included all adverbs inside an NP governed by the PP, while in the gold tree (Figure 4) both adverbs are attached to the PP. The modified parser output is shown in Figure 7.

5.3 Results for Converted Parser Output

We applied the conversion method described above to the original trees and the parser output for the sentences in the TIGER and the TüBa-D/Z test sets. Table 7 shows PARSEVAL and LA results for the modified trees, evaluating the converted parser output

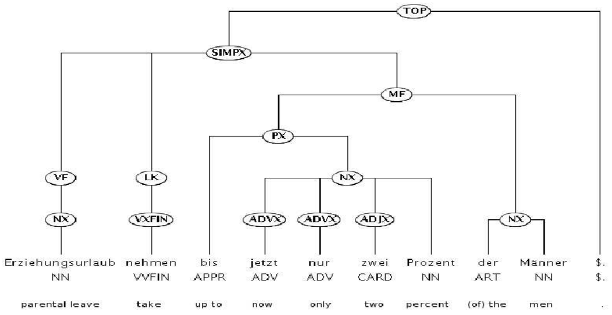

TOP
SIMPX
MF
PX
NX
VF
LK
NX
VXFIN
ADVX
ADVX
ADJX
NX
Erziehungsurlaub nehmen bis jetzt nur zwei Prozent der Männer $.
NN VVFIN APPR ADV ADV CARD NN ART NN $.
parental leave take up to now only two percent (of) the men .

Figure 6: Parser output (TüBa-D/Z grammar)

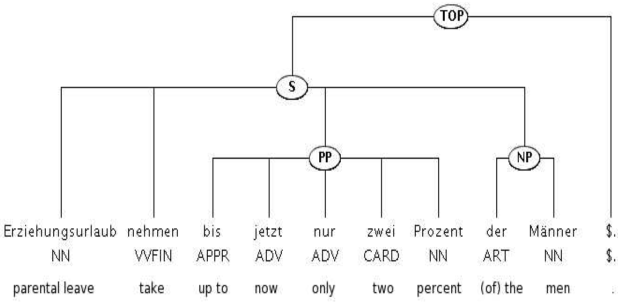

TOP
S
PP
NP
Erziehungsurlaub nehmen bis jetzt nur zwei Prozent der Männer $.
NN VVFIN APPR ADV ADV CARD NN ART NN $.
parental leave take up to now only two percent (of) the men .

Figure 7: Converted parser output (TüBa-D/Z)

for each treebank against the converted gold trees of the same treebank. Due to the resolved crossing branches in the TIGER treebank we also have some unary nodes in the TIGER test set. Their removal surprisingly improves both PARSEVAL and LA results. For the TüBa-D/Z all conversions lead to a decrease in precision and recall for the PARSEVAL metric. Converting the trees parsed by the TüBa-D/Z grammar to a TIGER-like format produces an f-score which is slightly lower than that for the TIGER trees. The same is true for the LA metric, but not to the same extent as for PARSEVAL. The LA metric also gives slightly better results for the original TIGER trees compared to the result for the unmodified TüBa-D/Z trees.

The constant decrease in PARSEVAL results for the modified trees is consistent with the results in Kübler et al. (2005), but our conclusions are slightly different. Our experiment shows that the TüBa-D/Z annotation scheme does not generally produce higher quality parser output, but that the PARSEVAL results are highly sensitive to the ratio of non-terminal/terminal nodes. However, the parser output for the grammar trained on the TüBa-D/Z yields a higher precision in the PARSEVAL metric against the TüBa-D/Z gold trees than the parser output of the TIGER grammar against the TIGER gold trees. For PARSEVAL recall, the TIGER grammar gives better results.

	EVALB			LA
	prec.	**recall**	**f-sco.**	**avg.**
TIGER	83.54	83.65	83.59	94.69
no Unary	84.33	84.48	84.41	94.83
TüBa-D/Z	92.59	89.79	91.17	94.23
TüBa-D/Z → TIGER				
no Top	92.38	88.76	90.53	93.93
no Unary	89.96	85.67	87.76	93.59
no Top + no U.	88.44	82.24	85.23	92.91
no Top + no U. + no NP in PP	87.15	79.52	83.16	92.47

Table 7: The impact of the conversion process on PARSEVAL and LA

6 Experiment III

In Experiment I and II we showed that the tree-based PARSEVAL metric is not a reliable measure for comparing the impact of different treebank annotation schemes on the quality of parser output and that the issue, whether German is harder to parse than English, remains undecided. In Experiment III we report a dependency-based evaluation and compare the results to the results of the other metrics.

6.1 Dependency-Based (DB) Evaluation

The dependency-based evaluation used in the experiments follows the method of Lin (1998) and Kübler and Telljohann (2002), converting the original treebank trees and the parser output into dependency relations of the form WORD POS HEAD. Functional labels have been omitted for parsing, therefore the dependencies do not comprise functional information. Figure 8 shows the original TIGER Treebank representation for the CFG tree in Figure 3. Square boxes denote grammatical functions. Figure 9 shows the dependency relations for the same tree, indicated by labelled arrows. Converted into a WORD POS HEAD triple format the dependency tree looks as follows (Table 8).

Following Lin (1998), our DB evaluation algorithm computes precision and recall:

- **Precision**: the percentage of dependency relationships in the parser output that are also

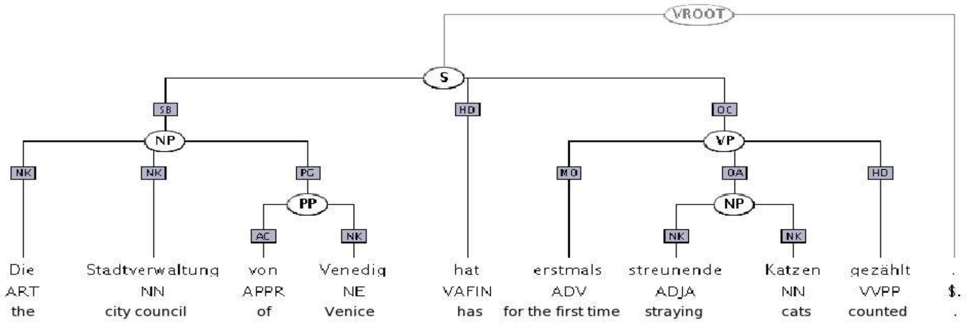
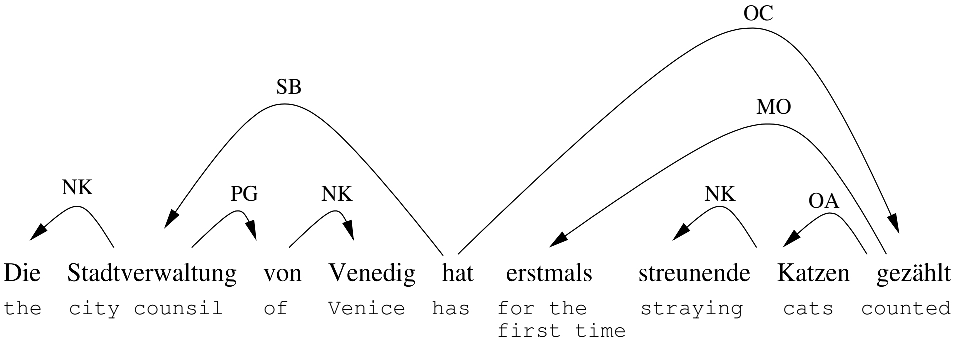


Figure 8: TIGER treebank representation for Figure 3



"For the first time the city counsil of Venice has counted straying cats."

Figure 9: Dependency relations for Figure 8

found in the gold triples

- **Recall**: the percentage of dependency relationships in the gold triples that are also found in the parser output triples.

WORD		**POS**	**HEAD**
Die	[the]	ART	Stadtverwaltung
Stadtverwaltung	[city counsil]	NN	hat
von	[of]	APPR	Stadtverwaltung
Venedig	[Venice]	NE	von
hat	[has]	VAFIN	-
erstmals	[for the first time]	ADV	gezählt
streunende	[straying]	ADJA	Katzen
Katzen	[cats]	NN	gezählt
gezählt	[counted]	VVPP	hat

Table 8: Dependency triples for Figure 9

We assessed the quality of the automatic conversion methodology by converting the 1024 original trees from each of our test sets into dependency relations, using the functional labels in the original trees to determine the dependencies. Topological fields in the TüBa-D/Z test set have been removed before extracting the dependency relationships.

We then removed all functional information from the trees and converted the stripped trees into dependencies, using heuristics to find the head. We evaluated the dependencies for the stripped gold trees against the dependencies for the original gold trees including functional labels and obtained an f-score of 99.64% for TIGER and 99.13% for the TüBa-D/Z dependencies. This shows that the conversion is reliable and not unduly biased to either the TIGER or TüBa-D/Z annotation schemes.

6.2 Experimental Setup

For Experiment III we used the same PCFG grammars and test sets as in Experiment II. Before extracting the dependency relationships we removed the topological fields in the TüBa-D/Z parser output. As shown in Section 6.1, this does not penalise the dependency-based evaluation results for the TüBa-D/Z. In contrast to Experiment II we used raw text as parser input instead of the gold POS tags, allowing a comparison with the gold tag results in Table 7.

6.3 Results

Table 9 shows the evaluation results for the three different evaluation metrics. For the DB evaluation the parser trained on the TIGER training set achieves about 7% higher results for precision and recall than the parser trained on the TüBa-D/Z. This result is clearly in contrast to the PARSEVAL scores, which show higher results for precision and recall for the TüBa-D/Z. But contrary to the PARSEVAL results on gold POS tags as parser input (Table 7), the gap between the results for TIGER and TüBa-D/Z is not as wide as before. PARSEVAL gives a labelled bracketing f-score of 81.12% (TIGER) and 85.47% (TüBa-D/Z) on raw text as parser input, while the results on gold POS tags are more distinctive with an f-score of 83.59% for TIGER and 91.17% for TüBa-D/Z. The LA results again give better scores to the TIGER parser output, this time the difference is more pronounced than for Experiment II (Table 7).

	Dependencies		PARSEVAL		LA
	Prec	Rec	Prec	Rec	Avg
TIGER	85.71	85.72	81.21	81.04	93.88
TüBa	76.64	76.63	87.24	83.77	92.58

Table 9: Parsing results for three evaluation metrics

The considerable difference between the results for the metrics raises the question which of the metrics is the most adequate for judging parser output quality across treebank encoding schemes.

7 Conclusions

In this paper we presented novel experiments assessing the validity of parsing results measured along different dimensions: the tree-based PARSEVAL metric, the string-based Leaf-Ancestor metric and a dependency-based evaluation. By inserting controlled errors into gold treebank trees and measuring the effects on parser evaluation results we gave new evidence for the downsides of PARSEVAL which, despite severe criticism, is still the standard measure for parser evaluation. We showed that PARSEVAL cannot be used to compare the output of PCFG parsers trained on different treebank annotation schemes, because the results correlate with the ratio of non-terminal/terminal nodes. Comparing two different annotation schemes, PARSEVAL consistently favours the one with the higher node ratio.

We examined the influence of treebank annotation schemes on unlexicalised PCFG parsing, and rejected the claim that the German TüBa-D/Z treebank is more appropriate for PCFG parsing than the German TIGER treebank and showed that converting the TüBa-D/Z trained parser output to a TIGER-like format leads to PARSEVAL results slightly worse than the ones for the TIGER treebank trained parser. Additional evidence comes from a dependency-based evaluation, showing that, for the output of the parser trained on the TIGER treebank, the mapping from the CFG trees to dependency relations yields better results than for the grammar trained on the TüBa-D/Z annotation scheme, even though PARSEVAL scores suggest that the TIGER-based parser output trees are substantial worse than TüBa-D/Z-based parser output trees.

We have shown that different treebank annotation schemes have a strong impact on parsing results for similar input data with similar (simulated) parser errors. Therefore the question whether a particular language is harder to parse than another language or not, can not be answered by comparing parsing results for parsers trained on treebanks with different annotation schemes. Comparing PARSEVAL-based parsing results for a parser trained on the TüBa-D/Z or TIGER to results achieved by a parser trained on the English Penn-II treebank (Marcus et al., 1994) does not provide conclusive evidence about the parsability of a particular language, because the results show a bias introduced by the combined effect of annotation scheme and evaluation metric. This means that the question whether German is harder to parse than English, is still undecided. A possible way forward is perhaps a dependency-based evaluation of TIGER/TüBa-D/Z with Penn-II trained grammars for 'similar' test and training sets and cross-treebank and -language controlled error insertion experiments. Even this is not entirely straightforward as it is not completely clear what constitutes 'similar' test/training sets across languages. We will attempt to pursue this in further research.

Acknowledgements

We would like to thank the anomymous reviewers for many helpful comments. This research has been supported by a Science Foundation Ireland grant 04|IN|I527.

References

Black, E., S. P. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. P. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. *In Proceedings DARPA Speech and Natural Language Workshop, Pacific Grove, CA*, pp. 306-311.

Brants, Sabine, and Silvia Hansen. 2002. Developments in the TIGER Annotation Scheme and their Realization in the Corpus. *In Proceedings of the Third Conference on Language Resources and Evaluation (LREC 2002)*, pp. 1643-1649 Las Palmas.

Briscoe, E. J., J. A. Carroll, and A. Copestake. 2002. Relational evaluation schemes. *In Proceedings Workshop 'Beyond Parseval - towards improved evaluation measures for parsing systems', 3rd International Conference on Language Resources and Evaluation*, pp. 4-38. Las Palmas, Canary Islands.

Carroll, J., E. Briscoe and A. Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. *In Proceedings of the 1st International Conference on Language Resources and Evaluation*, Granada, Spain. 447-454.

Dipper, S., T. Brants, W. Lezius, O. Plaehn, and G. Smith. 2001. The TIGER Treebank. *In Third Workshop on Linguistically Interpreted Corpora LINC-2001*, Leuven, Belgium.

Drach, Erich. 1937. *Grundgedanken der Deutschen Satzlehre.* Frankfurt/M.

Dubey, A., and F. Keller. 2003. Probabilistic parsing for German using sisterhead dependencies. *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Höhle, Tilman. 1998. Der Begriff "Mittelfeld", Anmerkungen über die Theorie der topologischen Felder. *In Akten des Siebten Internationalen Germansitenkongresses 1985*, pages 329-340, Göttingen, Germany.

Kübler, Sandra, and Heike Telljohann. 2002. Towards a Dependency-Oriented Evaluation for Partial Parsing. *In Proceedings of Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems (LREC 2002 Workshop)*, Las Palmas, Gran Canaria, June 2002.

Lin, Dekang. 1998. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 1998.

Kübler, Sandra. 2005. How Do Treebank Annotation Schemes Influence Parsing Results? Or How Not to Compare Apples And Oranges. *In Proceedings of FANLP 2005)*, Borovets, Bulgaria, September 2005.

Kübler, Sandra, Erhard Hinrichs, and Wolfgang Maier. 2006. Is it Really that Difficult to Parse German? *In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP 2006)*, Sydney, Australia, July 2006.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10.707-10 (translation of Russian original published in 1965).

Maier, Wolfgang. 2006. Annotation Schemes and their Influence on Parsing Results. *In Proceedings of the COLING/ACL 2006 Student Research Workshop)*, Sydney, Australia, July 2006.

Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, M. Ferguson, K. Katz and B. Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. *In Proceedings of the ARPA Human Language Technology Workshop*, Princeton, NJ.

Sampson, Geoffrey, and Anna Babarczy. 2003. A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, 9 (4):365-380.

Schmid, Helmut. 2000. LoPar: Design and Implementation. *Arbeitspapiere des Sonderforschungsbereiches 340, No. 149*, IMS Stuttgart, July 2000.

Schmid, Helmut. 2004. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. *In Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland.

Sekine, S. and M. J. Collins. 1997. The evalb software. http://nlp.cs.nyu.edu/evalb/

Skut, Wojciech, Brigitte Krann, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. *In Proceedings of ANLP 1997*, Washington, D.C.

Telljohann, Heike, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister. 2005. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z).* Seminar für Sprachwissenschaft, Universität Tübingen, Germany.

Schiller, Anne, Simone Teufel, and Christine Thielen. 1995. *Guidelines fr das Tagging deutscher Textcorpora mit STTS.* Technical Report, IMS-CL, University Stuttgart, 1995.

Semi-Markov Models for Sequence Segmentation

Qinfeng Shi
NICTA, Statistical Machine Learning
Australian National University
Canberra, 2601 ACT
qinfeng.shi@rsise.anu.edu.au

Yasemin Altun
Toyota Technological Institute
1427 E 60th St
Chicago, IL 60637
altun@tti-c.org

Alex Smola
NICTA, Statistical Machine Learning
Australian National University
Canberra, 2601 ACT
Alex.Smola@nicta.com.au

S. V. N. Vishwanathan
NICTA, Statistical Machine Learning
Australian National University
Canberra, 2601 ACT
SVN.Vishwanathan@nicta.com.au

Abstract

In this paper, we study the problem of automatically segmenting written text into paragraphs. This is inherently a sequence labeling problem, however, previous approaches ignore this dependency. We propose a novel approach for automatic paragraph segmentation, namely training Semi-Markov models discriminatively using a Max-Margin method. This method allows us to model the sequential nature of the problem and to incorporate features of a whole paragraph, such as *paragraph coherence* which cannot be used in previous models. Experimental evaluation on four text corpora shows improvement over the previous state-of-the art method on this task.

1 Introduction

In this paper, we study automatic paragraph segmentation (APS). This task is closely related to some well known problems such as text segmentation, discourse parsing, topic shift detection and is relevant for various important applications in speech-to-text and text-to-text tasks.

In speech-to-text applications, the output of a speech recognition system, such as the output of systems creating memos and documents for the Parliament House, is usually raw text without any punctuation or paragraph breaks. Clearly, such text requires paragraph segmentations. In text-to-text processing, such as summarization, the output text does not necessarily retain the correct paragraph structure and may require post-processing. There is psycholinguistic evidence as cited by Sporleder & Lapata (2004) showing that insertion of paragraph breaks could improve the readability. Moreover, it has been shown that different languages may have cross-linguistic variations in paragraph boundary placement (Zhu, 1999), which indicates that machine translation can also benefit from APS. APS can also recover the paragraph breaks that are often lost in the OCR applications.

There has been growing interest within the NLP community for APS in recent years. Previous methods such as Sporleder & Lapata (2004); Genzel (2005); Filippova & Strube (2006) treat the problem as a binary classification task, where each sentence is labeled as the beginning of a paragraph or not. They focus on the use of features, such as surface features, language modeling features and syntactic features. The effectiveness of features is investigated across languages and/or domains. However, these approaches ignore the inherent sequential nature of APS. Clearly, consecutive sentences within the same paragraph depend on each other. Moreover, paragraphs should exhibit certain properties such as coherence, which should be explored within an APS system. One cannot incorporate such properties/features when APS is treated as a binary classification problem. To overcome this limitation, we cast APS as a sequence prediction problem, where the performance can be significantly improved by optimizing the choice of labeling over whole sequences of sentences, rather than individual sentences.

Sequence prediction is one of the most promi-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 640–648, Prague, June 2007. ©2007 Association for Computational Linguistics

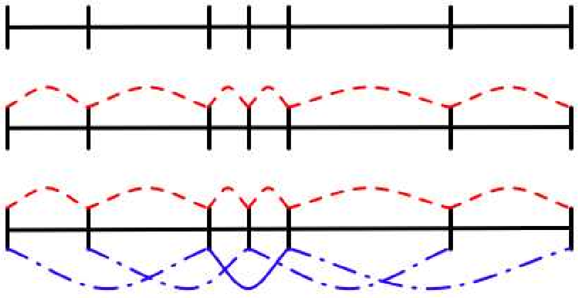


Figure 1: **Top**: sequence (horizontal line) with segment boundaries (vertical lines). This corresponds to a model where we estimate each segment boundary independently of all other boundaries. **Middle**: simple semi-Markov structure. The position of the segment boundaries only depends on the position of its neighbors, as denoted by the (red) dash arcs. **Bottom**: a more sophisticated semi-Markov structure, where each boundary depends on the position of two of its neighbors. This may occur, e.g., when the decision of where to place a boundary depends on the content of two adjacent segments. The longer range interaction is represented by the additional (blue) arcs.

nent examples of structured prediction. This problem is generally formalized such that there exists one variable for each observation in the sequence and the variables form a Markov chain (HMM). Segmentation of a sequence has been studied as a class of sequence prediction problems with common applications such as protein secondary structure prediction, Named Entity Recognition and segmentation of FAQ's. The exceptions to this approach are Sarawagi & Cohen (2004); Raetsch & Sonnenburg (2006), which show that Semi-Markov models (SMMs) (Janssen & Limnois, 1999), which are a variation of Markov models, are a natural formulation for sequence segmentation. The advantage of these models, depicted in Figure 1, is their ability to encode features that capture properties of a segment as a whole, which is not possible in an HMM model. In particular, these features can encode similarities between two sequence segments of arbitrary lengths, which can be very useful in tasks such as APS.

In this paper, we present a Semi-Markov model for APS and propose a max-margin training on these methods. This training method is a generalization of the Max-Margin methods for HMMs (Altun et al., 2003b) to SMMs. It follows the recent literature on discriminative learning of *structured prediction* (Lafferty et al., 2001; Collins, 2002; Altun et al., 2003a; Taskar et al., 2003). Our method inherits the advantages of discriminative techniques, namely the ability to encode arbitrary (overlapping) features and not making implausible conditional independence assumptions. It also has advantages of SMM models, namely the ability to encode features at segment level. We present a linear time inference algorithm for SMMs and outline the learning method. Experimental evaluation on datasets used previously on this task (Sporleder & Lapata, 2004) shows improvement over the state-of-the art methods on APS.

2 Modeling Sequence Segmentation

In sequence segmentation, our goal is to solve the estimation problem of finding a segmentation $y \in \mathcal{Y}$, given an observation sequence $x \in \mathcal{X}$. For example, in APS x can be a book which is a sequence of sentences. In a Semi-Markov model, there exists one variable for each subsequence of observations (i. e. multiple observations) and these variables form a Markov chain. This is opposed to an HMM where there exists one variable for each observation. More formally, in SMMs, $y \in \mathcal{Y}$ is a sequence of segment labelings $s_i = (b_i, l_i)$ where b_i is a non-negative integer denoting the beginning of the i^{th} segment which ends at position $b_{i+1} - 1$ and whose label is given by l_i (Sarawagi & Cohen, 2004). Since in APS the label of the segments is irrelevant, we represent each segment simply by the beginning position $y := \{b_i\}_{i=0}^{L-1}$ with the convention that $b_0 = 0$ and $b_L = N$ where N is the number of observations in x. Here, L denotes the number of segments in y. So the first segment is $[0, b_1)$, and the last segment is $[b_{L-1}, N)$, where $[a, b)$ denotes all the sentences from a to b inclusive a but exclusive b.

We cast this estimation problem as finding a discriminant function $F(x, y)$ such that for an observation sequence x we assign the segmentation that receives the best score with respect to F,

$$y^*(x) := \operatorname*{argmax}_{y \in \mathcal{Y}} F(x, y). \qquad (1)$$

As in many learning methods, we consider functions that are linear in some feature representation Φ,

$$F(x, y; w) = \langle w, \Phi(x, y)\rangle. \tag{2}$$

Here, $\Phi(x, y)$ is a feature map defined over the joint input/output space as detailed in Section 2.3.

2.1 Max-Margin Training

We now present a maximum margin training for predicting structured output variables, of which sequence segmentation is an instance. One of the advantages of this method is its ability to incorporate the cost function that the classifier is evaluated with. Let $\Delta(y, \bar{y})$ be the cost of predicting $\bar{y}$ instead of y. For instance, Δ is usually the 0-1 loss for binary and multiclass classification. However, in segmentation, this may be a more sophisticated function such as the symmetric difference of y and $\bar{y}$ as discussed in Section 2.2. Then, one can argue that optimizing a loss function that incorporates this cost can lead to better generalization properties. One can find a theoretical analysis of this approach in Tsochantaridis et al. (2004).

We follow the general framework of Tsochantaridis et al. (2004) and look for a hyperplane that separates the correct labeling y_i of each observation sequence x_i in our training set from all the incorrect labelings $\mathcal{Y} - y_i$ with some margin that depends on Δ additively [1]. In order to allow some outliers, we use slack variables ξ_i and maximize the minimum margin, $F(x_i, y_i) - \max_{y \in \mathcal{Y} - y_i} F(x_i, y)$, across training instances i. Equivalently,

$$\min_{w,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i \tag{3a}$$

$$\forall i, y \; \langle w, \Phi(x_i, y_i) - \Phi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i. \tag{3b}$$

To solve this optimization problem efficiently, one can investigate its dual given by

$$\begin{aligned} \min_{\alpha} \; & \frac{1}{2} \sum_{i,j,y,y'} \alpha_{iy}\alpha_{jy'} \left\langle \Phi(x_i, y), \Phi(x_j, y')\right\rangle \qquad (4) \\ & - \sum_{i,y} \Delta(y_i, y)\alpha_{iy} \\ \forall i, y \; & \sum_{y} \alpha_{iy} \leq C, \; \alpha_{iy} \geq 0. \end{aligned}$$

Here, there exists one parameter α_{iy} for each training instance x_i and its possible labeling $y \in \mathcal{Y}$. Solving this optimization problem presents a formidable challenge since $\mathcal{Y}$ generally scales exponentially with the number of variables within each variable y. This essentially makes it impossible to find an optimal solution via enumeration. Instead, one may use a column generation algorithm (Tsochantaridis et al., 2005) to find an approximate solution in polynomial time. The key idea is to find the most violated constraints (3b) for the current set of parameters and satisfy them up to some precision. In order to do this, one needs to find

$$\operatorname*{argmax}_{y \in \mathcal{Y}} \Delta(y_i, y) + \langle w, \Phi(x_i, y)\rangle, \tag{5}$$

which can usually be done via dynamic programming. As we shall see, this is an extension of the Viterbi algorithm for Semi Markov models.

Note that one can express the optimization and estimation problem in terms of kernels $k((x, y), (x', y')) := \langle \Phi(x, y), \Phi(x', y')\rangle$. We refer the reader to Tsochantaridis et al. (2005) for details.

To adapt the above framework to the segmentation setting, we need to address three issues: a) we need to specify a loss function Δ for segmentation, b) we need a suitable feature map Φ as defined in Section 2.3, and c) we need to find an algorithm to solve (5) efficiently. The max-margin training of SMMs was also presented in Raetsch & Sonnenburg (2006)

2.2 Cost Function

To measure the discrepancy between y and some alternative sequence segmentation y', we simply count the number of segment boundaries that have a) been missed and b) been wrongly added. Note that this definition allows for errors exceeding 100% - for

[1] There is an alternative formulation that is multiplicative in Δ. We prefer (3) due to computational efficiency reasons.

Algorithm 1 Max-Margin Training Algorithm

```
Input: data $x_i$, labels $y_i$, sample size $m$, tolerance $\epsilon$
Initialize $S_i = \emptyset$ for all $i$, and $w = 0$.
repeat
  for $i = 1$ to $m$ do
    $w = \sum_i \sum_{y \in S_i} \alpha_{iy} \Phi(x_i, y)$
    $y^* = \text{argmax}_{y \in \mathcal{Y}} \langle w, \Phi(x_i, y) \rangle + \Delta(y_i, y)$
    $\xi = \max(0, \max_{y \in S_i} \langle w, \Phi(x_i, y) \rangle + \Delta(y_i, y))$
    if $\langle w, \Phi(x_i, y^*) \rangle + \Delta(y_i, y) > \xi + \epsilon$ then
      Increase constraint set $S_i \leftarrow S_i \cup y^*$
      Optimize (4) wrt $\alpha_{iy}, \forall y \in S_i$.
    end if
  end for
until $S$ has not changed in this iteration
```

instance, if we were to place considerably more boundaries than can actually be found in a sequence.

The number of errors is given by the symmetric difference between y and y', when segmentations are viewed as sets. This can be written as

$$\begin{aligned}\Delta(y, y') &= |y| + |y'| - 2|y \cap y'| \\ &= |y| + \sum_{i=1}^{l'} \left[1 - 2\left\{b'_i \in y\right\}\right]. \quad (6)\end{aligned}$$

Here $|\cdot|$ denotes the cardinality of the set. Eq. (6) plays a vital role in solving (5), since it allows us to decompose the loss in y' into a constant and functions depending on the segment boundaries b'_i only. Note that in the case where we want to segment *and* label, we simply would need to check that the positions are accurate *and* that the labels of the segments match.

2.3 Feature Representation

SMMs can extract three kinds of features from the input/output pairs: a) node features, i. e. features that encode interactions between attributes of the observation sequence and the (label of a) *segment* (rather than the label of each observation as in HMM), b) features that encode interactions between neighboring labels along the sequence and c) edge features, i. e. features that encode properties of segments. The first two types of features are commonly used in other sequence models, such as HMMs and Conditional Random Fields (CRFs). The third feature type is specific to Semi-Markov models. In particular, these features can encode properties of a whole segment or similarities between two sequence segments of arbitrary lengths. The cost of this expressibility is simply a constant factor of the complexity of Markov models, if the maximum length of a segment is bounded. This type of features are particularly useful in the face of sparse data.

As in HMMs, we assume stationarity in our model and sum over the features of each segment to get $\Phi(x, y)$. Then, Φ corresponding to models of the middle structure given in Figure 1 is given by

$$\Phi(x, \bar{y}) := (\Phi_0, \sum_{i=1}^{\bar{l}-1} \Phi_1(\bar{n}_i, x), \sum_{i=1}^{\bar{l}} \Phi_2(\bar{b}_{i-1}, \bar{b}_i, x)).$$

We let $\Phi_0 = \bar{l} - 1$, the number of segments. The node features Φ_1 capture the dependency of the current segment boundary to the observations, whereas the edge features Φ_2 represent the dependency of the current segment to the observations. To model the bottom structure in Figure 1, one can design features that represent the dependency of the current segment to its adjacent segments as well as the observations, $\Phi_3(x, b_{i-2}, b_{i-1}, b_i)$. The specific choices of the feature map Φ are presented in Section 3.

2.4 Column Generation on SMMs

Tractability of Algorithm 1 depends on the existence of an efficient algorithm that finds the most violated constraint (3b) via (5). Both the cost function of Section 2.2 and the feature representation of Section 2.3 are defined over a short sequence of segment boundaries. Therefore, using the Markovian property, one can perform the above maximization step efficiently via a dynamic programming algorithm. This is a simple extension of the Viterbi algorithm. The inference given by (1) can be performed using the same algorithm, setting Δ to a constant function.

We first state the dynamic programming recursion for $F + \Delta$ in its generality. We then give the pseudocode for $\Phi_3 = \emptyset$.

Denote by $T(t_-, t_+; x)$ the largest value of $\Delta(y, p) + F(x, p)$ for any *partial* segmentation p that starts at position 0 and which ends with the segment $[t_-, t_+)$. Moreover, let M be a upper bound on the

Algorithm 2 Column Generation

```
Input: sequence $x$, segmentation $y$, max-length of a segment $M$
Output: score $s$, segment boundaries $y'$
Initialize vectors $T \in \mathbb{R}^m$ and $R \in \mathcal{Y}^m$ to 0
for $i = 1$ to $l$ do
    $R_i = \operatorname*{argmax}_{\max(0,i-M)\le j<i} T_j + g(j,i)$
    $T_i = T_{R_i} + g(R_i, i)$
end for
$s = T_m + |y|$
$y' = \{m\}$
repeat
    $i = y'_{\text{first}}$
    $y' \leftarrow \{R_i, y'\}$
until $i = 0$
```

length of a segment. The recursive step of the dynamic program is given by

$$\begin{aligned} T(t_-, t_+; x) &= \max_{\max(0,t_- - M)\le k<t_-} T(k, t_-; x) \\ &+ g(k, t_-, t_+) \end{aligned}$$

where we defined the increment $g(k, t_-, t_+)$ as

$$\langle \Phi_0(x), \Phi_1(x, t_+), \Phi_2(x, t_-, t_+), \Phi_3(x, k, t_-, t_+), w\rangle \\ +1 - 2\{(t_-, t_+) \in y\}$$

where by convention $T(i, i') = -\infty$ if $i < 0$ for all labels. Since T needs to be computed for all values of $t_+ - M \le t_- < t_+$, we need to compute $O(|x|M)$ many values, each of which requires an optimization over M possible values. That is, storage requirements are $O(|x|M)$, whereas the computation scales with $O(|x|M^2)$. If we have a good bound on the maximal sequence length, this can be dealt with efficiently. Finally, the recursion is set up by $T(0, 0, x) = |y|$.

See Algorithm 2 for pseudocode, when $\Phi_3 = \emptyset$. The segmentation corresponding to (5) is found by constructing the path traversed by the argument of the $\max$ operation generating T.

3 Features

We now specify the features described in Section 2.3 for APS. Note that the second type of features do not exist for APS since we ignore the labelings of segments.

3.1 Node Features Φ_1

Node features $\Phi_1(b_j, x)$ represent the information of the current segment boundary and some attributes of the observations around it (which we define as the current, preceding and successive sentences). These are sentence level features, which we adapt from Genzel (2005) and Sporleder & Lapata (2004) [2]. For the b_jth sentence, $x(b_j)$, we use the following features

- Length of $x(b_j)$.
- Relative Position of $x(b_j)$.
- Final punctuation of $x(b_j)$.
- Number of capitalized words in $x(b_j)$.
- Word Overlap of $x(b_j)$ with the next one

  $$W_{over}(x(b_j), x(b_j+1)) = \frac{2 \mid x(b_j) \cap x(b_j+1) \mid}{\mid x(b_j) \mid + \mid x(b_j+1) \mid}.$$

- First word of $x(b_j)$.
- Bag Of Words (BOW) features: Let the bag of words of a set of sentences S be

  $$B(S) = (c_0, c_1, ..., c_i, ..., c_{N-1}),$$

  where N is the size of the dictionary and c_i is the frequency of word i in S.

  - BOW of $x(b_j)$, $B(\{x(b_j)\})$
  - BOW of $x(b_j)$ and the previous sentence $B(\{x(b_j-1), x(b_j)\})$
  - BOW of $x(b_j)$ and the succeeding sentence $B(\{x(b_j), x(b_j+1)\})$
  - The inner product of the two items above

- Cosine Similarity of $x(b_j)$ and the previous sentence

  $$CS(x(b_j-1), x(b_j)) \\ = \frac{\langle B(x(b_j-1)), B(x(b_j))\rangle}{\mid B(x(b_j-1)) \mid \times \mid B(x(b_j)) \mid}$$

[2] Due to space limitations, we omit the motivations for these features and refer the reader to the literature cited above.

- Shannon's Entropy of $x(b_j)$ computed by using a language model as described in Genzel & Charniak (2003).
- Quotes(Q_p, Q_c, Q_i). Q_p and Q_c are the number of pairs of quotes in the previous(Num_p) and current sentence (Num_c), $Q_p = 0.5 \times Num_p$ and $Q_c = 0.5 \times Num_c$.

3.1.1 Edge Features Φ_2

Below is the set of features $\Phi_2(b_j, b_{j+1}, x)$ encoding information about the current segment. These features represent the power of the Semi-Markov models. Note that Φ_3 features also belong to edge features category. In this paper, we did not use Φ_3 feature due to computational issues.

- Length of The Paragraph: This feature expresses the assumption that one would want to have a balance across the lengths of the paragraphs assigned to a text. Very long and very short paragraphs should be uncommon.
- Cosine Similarity of the current paragraph and neighboring sentences: Ideally, one would like to measure the similarity of two consecutive paragraphs and search for a segmentation that assigns low similarity scores (in order to facilitate changes in the content). This can be encoded using $\Phi_3(x, b_{j-1}, b_j, b_{j+1})$ features. When such features are computationally expensive, one can measure the similarity of the current paragraph with the preceding sentence as

  $$\begin{aligned} &CS(P, x(b_j - 1)) \\ &= \frac{\langle BOW(P), BOW(x(b_j - 1))\rangle}{\mid BOW(P) \mid \times \mid BOW(x(b_j - 1)) \mid} \end{aligned}$$

  where P is the set of sentences in the current paragraph, $[b_j, b_{j+1})$. A similar feature is used for $CS(P, x(b_{j+1}))$.
- Shannon's Entropy of the Paragraph: The motivation for including features encoding the entropy of the sentences is the observation that the entropy of paragraph initial sentences is lower than the others (Genzel & Charniak, 2003). The motivation for including features encoding the entropy of the paragraphs, on the other hand, is that the entropy rate should remain more or less constant across paragraphs, especially for long texts like books. We ignore the sentence boundaries and use the same technique that we use to compute the entropy of a sentence.

3.2 Feature Rescaling

Most of the features described above are binary. There are also some features such as the entropy whose value could be very large. We rescale all the non-binary valued features so that they do not override the effect of the binary features. The scaling is performed as follows:

$$u_{new} = \frac{u - \min(u)}{\max(u) - \min(u)}$$

where u_{new} is the new feature and u is the old feature. $\min(u)$ is the minimum of u, and $\max(u)$ is the maximum of u. An exception to this is the rescaling of BOW features which is given by

$$B(x(b_j))_{new} = B(x(b_j))/\langle B(x(b_j)), B(x(b_j))\rangle$$

$\langle ., . \rangle$ denotes the inner product.

4 Experiments

We collected four sets of data for our experiments. The first corpus, which we call SB, consists of manually annotated text from the book *The Adventures of Bruce-Partington Plans* by Arthur Conan-Doyle. The second corpus, which we call SA, again consists of manually annotated text but from 10 different books by Conan-Doyle. Our third corpus consists of German (GER) and English (ENG) texts. The German data consisting of 12 German novels was used by Sporleder & Lapata (2006). This data uses automatically assigned paragraph boundaries, with the labeling error expected to be around 10%. The English data contains 12 well known English books from Project Gutenberg (`http://www.gutenberg.org/wiki/Main_Page`). For this dataset the paragraph boundaries were marked manually.

All corpora were approximately split into training (72%), development (21%), and test set (7%) (see Table 1). The table also reports the accuracy of the baseline classifier, denoted as BASE, which either labels all sentences as paragraph boundaries or

Table 1: Number of sentences and % accuracy of the baseline classifier (BASE) on various datasets used in our experiments.

	TOTAL	TRAIN	DEV	TEST	BASE
SB	59,870	43,678	12,174	3,839	53.70
SA	69,369	50,680	14,204	4,485	58.62
ENG	123,261	88,808	25,864	8,589	63.41
GER	370,990	340,416	98,610	31,964	62.10

Table 2: Test results on ENG and GER data after model selection.

DATASET	ALGO.	ACC.	REC.	PREC.	F_1
ENG	SMM	**75.61**	**46.67**	**77.78**	**58.33**
	SVM	58.54	26.67	40.00	32.00
	BT	65.85	33.33	55.56	41.67
GER	SMM	70.56	46.81	65.67	54.66
	SVM	39.92	**100.00**	38.68	55.79
	BT	**72.58**	54.26	**67.11**	**60.00**

non-boundaries, choosing whichever scheme yields a better accuracy.

We evaluate our system using accuracy, precision, recall, and the F_1-score given by $(2 \times Precision \times Recall)/(Precision + Recall)$ and compare our results to Sporleder & Lapata (2006) who used BoosTexter (Schapire & Singer, 2000) as a learning algorithm. To the best of our knowledge, BoosTexter (henceforth called BT) is the leading method published for this task so far. In order to evaluate the importance of the edge features and the resultant large-margin constraint, we also compare against a standard binary Support Vector Machine (SVM) which uses node features alone to predict whether each sentence is the beginning of a paragraph or not. For a fair comparison, all classifiers used the linear kernel and the same set of node features.

We perform model selection for all three algorithms by choosing the parameter values that achieve the best F_1-score on the development set. For both the SVM as well as our algorithm, SMM, we tune the parameter C (see (3a)) which measures the trade-off between training error and margin. For BT, we tune the number of Boosting iterations, denoted by N.

4.1 Results

In our first experiment, we compare the performance of our algorithm, SMM, on the English and German corpus to a standard SVM and BoosTexter. We report these result in Table 2. Our algorithm achieves the best F_1-score on the ENG corpus. SMM performs very competitively on the GER corpus, achieving accuracies close to those of BT.

We observed a large discrepancy between the performance of our algorithm on the development and the test datasets. The situation is similar for both SVM and BT. For instance, BT when trained on the ENG corpora, achieves an optimal F_1-score of 18.67% after $N = 100$ iterations. For the same N value, the test performance is 41.67%. We conjecture that this discrepancy is because the books that we use for training and test are written by different authors. While there is some generic information about when to insert a paragraph break, it is often subjective and part of the authors style. To test this hypothesis, we performed experiments on the SA and SB corpus, and present results in Table 3. Indeed, the F_1-scores obtained on the development and test corpus closely match for text drawn from the same book (whilst exhibiting better overall performance), differs slightly for text drawn from different books by the same author, and has a large deviation for the GER and ENG corpus.

Table 3: Comparison on various ENG datasets.

DATASET	ACC.	REC.	PREC.	F_1-SCORE
SB (DEV)	92.81	86.44	92.73	89.47
SB (TEST)	96.30	96.00	96.00	96.00
SA (DEV)	82.24	61.11	82.38	70.17
SA (TEST)	81.03	79.17	76.00	77.55
ENG (DEV)	69.84	18.46	78.63	29.90
ENG (TEST)	75.61	46.67	77.78	58.33

There is one extra degree of freedom that we can optimize in our model, namely the offset, i. e. the weight assigned to the constant feature Φ_0. After fixing all the parameters as described above, we vary the value of the offset parameter and pick the value that gives the F_1-score on the development data. We choose to use F_1-score, since it is the error measure that we care about. Although this extra optimization leads to better F_1-score in German (69.35% as opposed to 54.66% where there is no extra tuning of the offset), it results in a decrease of the F_1-score in English (52.28% as opposed to 58.33%). These results are reported in Table 4. We found that the difference of the F_1-score of tuning and not tuning the threshold on the development set was not a good indicator on the usefulness of this extra parameter. We are now investigating other properties, such as variance on the development data, to see if the tuning of the threshold can be used for better APS systems.

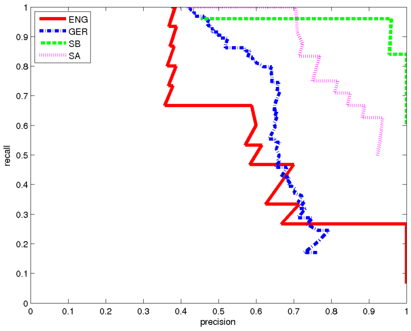


Figure 2: Precision-recall curves

Figure 2 plots the precision-recall curve obtained on various datasets. As can be seen the performance of our algorithm on the SB dataset is close to optimum, whilst it degrades slightly on the SA dataset, and substantially on the ENG and GER datasets. This further confirms our hypothesis that our algorithm excels in capturing stylistic elements from a single author, but suffers slightly when trained to identify generic stylistic elements. We note that this is not a weakness of our approach alone. In fact, all the other learning algorithms also suffer from this shortcoming.

Table 4: Performance on ENG test set tuning the offset for best F_1-score on ENG development set.

DATASET	ACC.	REC.	PREC.	F_1-SCORE
ENG	75.61	46.67	77.78	58.33
ENG +Φ_0	39.02	93.33	36.84	52.28
GER	70.56	46.81	65.67	54.66
GER + Φ_0	75.40	73.40	65.71	69.35

5 Conclusion

We presented a competitive algorithm for paragraph segmentation which uses the ideas from large margin classifiers and graphical models to extend the semi-Markov formalism to the large margin case. We obtain an efficient dynamic programming formulation for segmentation which works in linear time in the length of the sequence. Experimental evaluation shows that our algorithm is competitive when compared to the state-of-the-art methods.

As future work, we plan on implementing Φ_3 features in order to perform an accuracy/time analysis. By defining appropriate features, we can use our method immediately for text and discourse segmentation. It would be interesting to compare this method to Latent Semantic Analysis approaches for text segmentation as studied for example in Bestgen (2006) and the references thereof.

References

Altun, Y., Hofmann, T., & Johnson, M. (2003a). Discriminative Learning for Label Sequences via Boosting. In *In Proceedings of NIPS 2003*.

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003b). Hidden markov support vector machines. In *International Conference on Machine Learning*.

Bestgen, Y. (2006). Improving text segmentation using latent semantic analysis: A reanalysis of choi, wiemer-hastings, and moore (2001). *Computational Linguistics*, *32*, 5–12.

Collins, M. (2002). Discriminative training methods for hidden markov models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Filippova, K., & Strube, M. (2006). Using linguistically motivated features for paragraph segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Genzel, D. (2005). A paragraph boundary detection system. In *Proceedings of the Conference on Computational Linguistics and Intelligent Text Processing*.

Genzel, D., & Charniak, E. (2003). Variation of entropy and parse tree of sentences as a function of the sentence number. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Janssen, J., & Limnois, N. (1999). *Semi-markov models and applications*. Kluwer Academic.

Lafferty, J. D., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *18th International Conference on Machine Learning ICML*.

Raetsch, G., & Sonnenburg, S. (2006). Large scale hidden Semi-Markov SVMs for gene structure prediction. In *In Proceedings of NIPS 2006*.

Sarawagi, S., & Cohen, W. (2004). Semi-Markov Conditional Random Fields for Information Extraction. In *Advances in Neural Information Processing Systems (NIPS)*.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Sporleder, C., & Lapata, M. (2004). Automatic paragraph identification: A study across languages and domains. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sporleder, C., & Lapata, M. (2006). Broad coverage paragraph segmentation across languages and domains. *ACM Trans. Speech Lang. Process.*, *3*(2), 1–35.

Taskar, B., Guestrin, C., & Koller, D. (2003). Maxmargin markov networks. In S. Thrun, L. Saul, & B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Twenty-first international conference on Machine learning*. New York, NY, USA: ACM Press. ISBN 1-58113-828-5.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*.

Zhu, C. (1999). Ut once more: The sentence as the key functional unit of translation. *Meta*, *44(3)*, 429–447.

A Graph-based Approach to Named Entity Categorization in Wikipedia Using Conditional Random Fields

Yotaro Watanabe, Masayuki Asahara and Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192, Japan
{yotaro-w,masayu-a,matsu}@is.naist.jp

Abstract

This paper presents a method for categorizing named entities in Wikipedia. In Wikipedia, an anchor text is glossed in a linked HTML text. We formalize named entity categorization as a task of categorizing anchor texts with linked HTML texts which glosses a named entity. Using this representation, we introduce a graph structure in which anchor texts are regarded as nodes. In order to incorporate HTML structure on the graph, three types of cliques are defined based on the HTML tree structure. We propose a method with Conditional Random Fields (CRFs) to categorize the nodes on the graph. Since the defined graph may include cycles, the exact inference of CRFs is computationally expensive. We introduce an approximate inference method using Tree-based Reparameterization (TRP) to reduce computational cost. In experiments, our proposed model obtained significant improvements compare to baseline models that use Support Vector Machines.

1 Introduction

Named and Numeric Entities (NEs) refer to proper nouns (e.g. PERSON, LOCATION and ORGANIZATION), time expressions, date expressions and so on. Since a large number of NEs exist in the world, unknown expressions appear frequently in texts, and they become hindrance to real-world text analysis. To cope with the problem, one effective ways to add a large number of NEs to gazetteers.

In recent years, NE extraction has been performed with machine learning based methods. However, such methods cannot cover all of NEs in texts. Therefore, it is necessary to extract NEs from existing resources and use them to identify more NEs. There are many useful resources on the Web. We focus on Wikipedia[1] as the resource for acquiring NEs. Wikipedia is a free multilingual online encyclopedia and a rapidly growing resource. In Wikipedia, a large number of NEs are described in titles of articles with useful information such as HTML tree structures and categories. Each article links to other related articles. According to these characteristics, they could be an appropriate resource for extracting NEs.

Since a specific entity or concept is glossed in a Wikipedia article, we can regard the NE extraction problem as a document classification problem of the Wikipedia article. In traditional approaches for document classification, in many cases, documents are classified independently. However, the Wikipedia articles are hypertexts and they have a rich structure that is useful for categorization. For example, hyperlinked mentions (we call them **anchor texts**) which are enumerated in a list tend to refer to the articles that describe other NEs belonging to the same class. It is expected that improved NE categorization is accomplished by capturing such dependencies.

We structure anchor texts and dependencies between them into a graph, and train graph-based CRFs to obtain probabilistic models to estimate categories for NEs in Wikipedia.

So far, several statistical models that can cap-

[1] http://wikipedia.org/

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 649–657, Prague, June 2007. ©2007 Association for Computational Linguistics

ture dependencies between examples have been proposed. There are two types of classification methods that can capture dependencies: iterative classification methods (Neville and Jensen, 2000; Lu and Getoor, 2003b) and collective classification methods (Getoor et al., 2001; Taskar et al., 2002). In this paper, we use Conditional Random Fields (CRFs) (Lafferty et al., 2001) for NE categorization in Wikipedia.

The rest of the paper is structured as follows. Section 2 describes the general framework of CRFs. Section 3 describes a graph-based CRFs for NE categorization in Wikipedia. In section 4, we show the experimental results. Section 5 describes related work. We conclude in section 6.

2 Conditional Random Fields

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are undirected graphical models that give a conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$ in a form of exponential model.

CRFs are formalized as follows. Let $\mathcal{G} = \{V, E\}$ be an undirected graph over random variables $\boldsymbol{y}$ and $\boldsymbol{x}$, where V is a set of vertices, and E is a set of edges in the graph $\mathcal{G}$. When a set of cliques $C = \{\{\boldsymbol{y}_c, \boldsymbol{x}_c\}\}$ are given, CRFs define the conditional probability of a state assignment given an observation set.

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{c \in \boldsymbol{C}} \Phi(\boldsymbol{x}_c, \boldsymbol{y}_c) \qquad (1)$$

where $\Phi(\boldsymbol{x}_c, \boldsymbol{y}_c)$ is a potential function defined over cliques, and $Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \prod_{c \in \boldsymbol{C}} \Phi(\boldsymbol{x}_c, \boldsymbol{y}_c)$ is the partition function.

The potentials are factorized according to the set of features $\{f_k\}$.

$$\Phi(\boldsymbol{x}_c, \boldsymbol{y}_c) = \exp\left(\sum_k \lambda_k f_k(\boldsymbol{x}_c, \boldsymbol{y}_c)\right) \qquad (2)$$

where $F = \{f_1, ..., f_K\}$ are feature functions on the cliques, $\Lambda = \{\lambda_1, ..., \lambda_K \in \mathcal{R}\}$ are the model parameters. The parameters Λ are estimated iterative scaling or quasi-Newton method from labeled data.

The original paper (Lafferty et al., 2001) focused on linear-chain CRFs, and applied them to part-of-speech tagging problem. McCallum et al. (2003), Sutton et al (2004) proposed Dynamic Conditional Random Fields (DCRFs), the generalization of linear-chain CRFs, that have complex graph structure (include cycles). Since DCRFs model structure contains cycles, it is necessary to use approximate inference methods to calculate marginal probability. Tree-based Reparameterization (TRP) (Wainwright et al., 2003), a schedule for loopy belief propagation, is used for approximate inference in these papers.

3 Graph-based CRFs for NE Categorization in Wikipedia

In this section we describe how to apply CRFs for NE categorization in Wikipedia.

Each Wikipedia article describes a specific entity or concept by a heading word, a definition, and one or more categories. One possible approach is to classify each NE described in an article into an appropriate category by exploiting the definition of the article. This process can be done one by one without considering the relationship with other articles.

On the other hand, articles in Wikipedia are semi-structured texts. Especially lists `(<UL> or <OL>)` and tables `(<TABLE>)` have an important characteristics, that is, occurrence of elements in them have some sort of dependencies. Structural characteristics, such as lists `(<UL> or <OL>)` or tables `(<TABLE>)`, are useful becase their elements have some sort of dependencies.

Figure 2 shows an example of an HTML segment and the corresponding tree structure. The first anchor texts in each list tag (`<LI>`) tend to be in the same NE category. Such characteristics are useful feature for the categorization task. In this paper we focus on lists which appear frequently in Wikipedia.

Furthermore, there are anchor texts in articles. Anchor texts are glossed entity or concept described with links to other pages. With this in mind, our NE categorization problem can be regarded as NE category labeling problem for anchor texts in articles. Exploiting dependencies of anchor texts that are induced by the HTML structure is expected to improve categorization performance.

We use CRFs for categorization in which anchor texts correspond to random variables V in $\mathcal{G}$ and de-

Sibling $E_S = \{(v_i^{\mathcal{T}}, v_j^{\mathcal{T}}) | v_i^{\mathcal{T}}, v_j^{\mathcal{T}} \in V^{\mathcal{T}},\ d(v_i^{\mathcal{T}},\ ca(v_i^{\mathcal{T}},\ v_j^{\mathcal{T}})) = d(v_j^{\mathcal{T}},\ ca(v_i^{\mathcal{T}},\ v_j^{\mathcal{T}})) = 1,\ v_j^{\mathcal{T}} = ch(pa(v_j^{\mathcal{T}}, 1), k),$
$v_i^{\mathcal{T}} = ch(pa(v_i^{\mathcal{T}}, 1),\ \max\{l | l < k\})\}$

Cousin $E_C = \{(v_i^{\mathcal{T}}, v_j^{\mathcal{T}}) | v_i^{\mathcal{T}},\ v_j^{\mathcal{T}} \in V^{\mathcal{T}}, d(v_i^{\mathcal{T}},\ ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})) = d(v_j^{\mathcal{T}},\ ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})) \geq 2,\ v_i^{\mathcal{T}} = ch(pa(v_i^{\mathcal{T}}), k),$
$v_j^{\mathcal{T}} = ch(pa(v_j^{\mathcal{T}}), k),\ pa(v_j^{\mathcal{T}}, d(v_j^{\mathcal{T}}, ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})) - 1) = ch(pa(v_j^{\mathcal{T}}, d(v_j^{\mathcal{T}}, ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}}))), k),$
$pa(v_i^{\mathcal{T}}, d(v_i^{\mathcal{T}}, ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})) - 1) = ch(pa(v_i^{\mathcal{T}}, ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})), \max\{l | l < k\})\}$

Relative $E_R = \{(v_i^{\mathcal{T}}, v_j^{\mathcal{T}}) | v_i^{\mathcal{T}}, v_j^{\mathcal{T}} \in V^{\mathcal{T}}, d(v_i^{\mathcal{T}}, ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})) = 1, d(v_j^{\mathcal{T}}, ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})) = 3,$
$pa(v_j^{\mathcal{T}}, 2) = ch(pa(v_j^{\mathcal{T}}, 3), k), v_i^{\mathcal{T}} = ch(pa(v_i^{\mathcal{T}}, 1), \max\{l | l < k\})\}$

Figure 1: The definitions of sibling, cousin and relative cliques, where E_S, E_C, E_R correspond to sets which consist of anchor text pairs that have sibling, cousin and relative relations respectively.

pendencies between anchor texts are treated as edges E in $\mathcal{G}$. In the next section, we describe the concrete way to construct graphs.

3.1 Constructing a graph from an HTML tree

An HTML document is an ordered tree. We define a graph $\mathcal{G} = (V^{\mathcal{G}}, E^{\mathcal{G}})$ on an HTML tree $\mathcal{T}^{HTML} = (V^{\mathcal{T}}, E^{\mathcal{T}})$: the vertices $V^{\mathcal{G}}$ are anchor texts in the HTML text; the edges E are limited to cliques of Sibling, Cousin, and Relative, which we will describe later in the section. These cliques are intended to encode a NE label dependency between anchor texts where the two NEs tend to be in the same or related class, or one NE affects the other NE label.

Let us consider dependent anchor text pairs in Figure 2. First, “Dillard & Clark” and “country rock” have a sibling relation over the tree structure, and appearing the same element of the list. The latter element in this relation tends to be an attribute or a concept of the other element in the relation. Second, “Dillard & Clark” and “Carpenters” have a cousin relation over the tree structure, and they tend to have a common attribute such as “Artist”. The elements in this relation tend to belong to the same class. Third, “Carpenters” and “Karen Carpenter” have a relation in which “Karen Carpenter” is a sibling’s grandchild in relation to “Carpenters” over the tree structure. The latter elements in this relation tends to be a constituent part of the other element in the relation. We can say that the model can capture dependencies by dealing with anchor texts that depend on each other as cliques. Based on the observations as above, we treat a pair of anchor texts as cliques which satisfy the condtions in Figure 1.

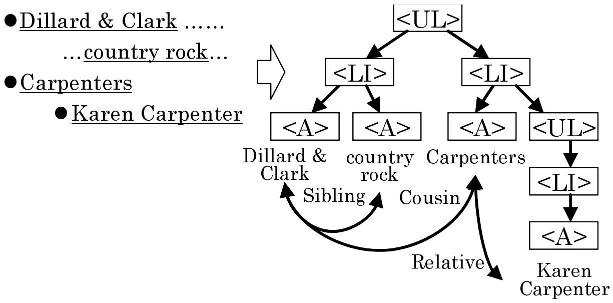


Figure 2: Correspondence between tree structure and defined cliques.

Now, we define the three sorts of edges given an HTML tree. Consider an HTML tree $\mathcal{T}^{HTML} = (V^{\mathcal{T}}, E^{\mathcal{T}})$, where $V^{\mathcal{T}}$ and $E^{\mathcal{T}}$ are nodes and edges over the tree. Let $d(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})$ be the number of edges between $v_i^{\mathcal{T}}$ and $v_j^{\mathcal{T}}$ where $v_i^{\mathcal{T}}, v_j^{\mathcal{T}} \in V^{\mathcal{T}}$, $pa(v_i^{\mathcal{T}}, k)$ be k-th generation ancestor of $v_i^{\mathcal{T}}$, $ch(v_i^{\mathcal{T}}, k)$ be $v_i^{\mathcal{T}}$’s k-th child, $ca(v_i^{\mathcal{T}}, v_j^{\mathcal{T}})$ be a common ancestor of $v_i^{\mathcal{T}}, v_j^{\mathcal{T}} \in V^{\mathcal{T}}$. Precise definitions of cliques, namely Sibling, Cousin, and Relative, are given in Figure 1. A set of cliques used in our graph-based CRFs are edges defined in Figure 1 and vertices, i.e. $C = E_S \cup E_C \cup E_R \cup V$. Note that they are restricted to pairs of the nearest vertices to keep the graph simple.

3.2 Model

We introduce potential functions for cliques to define conditional probability distribution over CRFs. Conditional distribution over label set $\boldsymbol{y}$ given observation set $\boldsymbol{x}$ is defined as:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}\left(\prod_{(v_i,v_j)\in E_S\cup E_C\cup E_R}\Phi_{SCR}(y_i,y_j)\right)\left(\prod_{v_i\in V}\Phi_V(y_i,\boldsymbol{x})\right) \quad (3)$$

where $\Phi_{SCR}(y_i, y_j)$ is the potential over sibling, cousin and relative edges, $\Phi_V(y_i, \boldsymbol{x})$ is the potential over the nodes, and $Z(\boldsymbol{x})$ is the partition function. The potentials $\Phi_{SCR}(y_i, y_j)$ and $\Phi_V(y_i, \boldsymbol{x})$ factorize according to the features f_k and weights λ_k as:

$$\Phi_{SCR}\left(y_i,y_j\right) = \exp\left(\sum_k \lambda_k f_k(y_i,y_j)\right) \quad (4)$$

$$\Phi_V\left(y_i,\boldsymbol{x}\right) = \exp\left(\sum_{k'} \lambda_{k'} f_{k'}(y_i,\boldsymbol{x})\right) \quad (5)$$

$f_k(y_i, y_j)$ captures co-occurrences between labels, where $k \in \{(y_i, y_j)|\mathcal{Y} \times \mathcal{Y}\}$ corresponds to the particular element of the Cartesian product of the label set $\mathcal{Y}$. $f_{k'}(y_i, \boldsymbol{x})$ captures co-occurrences between label $y_i \in \mathcal{Y}$ and observation features, where k' corresponds to the particular element of the label set and observed features.

The weights of a CRF, $\Lambda = \{\lambda_k, \ldots, \lambda_{k'}, \ldots\}$ are estimated to maximize the conditional log-likelihood of the graph in a training dataset $\mathcal{D} = \{\langle \boldsymbol{x}^{(1)}, y^{(1)}\rangle, \langle \boldsymbol{x}^{(2)}, y^{(2)}\rangle, \ldots, \langle \boldsymbol{x}^{(N)}, y^{(N)}\rangle\}$ The log-likelihood function can be defined as follows:

$$\begin{aligned}\mathcal{L}_\lambda = \sum_{d=1}^{N}[&\sum_{(v_i,v_j)\in E_S^{(d)}\cup E_C^{(d)}\cup E_R^{(d)}} \sum_k \lambda_k f_k(y_i,y_j) \\ &+ \sum_{v_i\in V^{(d)}}\sum_{k'}\lambda_{k'} f_{k'}(y_i,\boldsymbol{x}^{(d)}) - logZ(\boldsymbol{x}^{(d)})] \\ &- \sum_k \frac{\lambda_k^2}{2\sigma^2} - \sum_{k'}\frac{\lambda_{k'}^2}{2\sigma^2} \quad (6)\end{aligned}$$

where the last two terms are due to the Gaussian prior (Chen and Rosenfeld, 1999) used to reduce overfitting. Quasi-Newton methods, such as L-BFGS (Liu and Nocedal, 1989) can be used for maximizing the function.

3.3 Tree-based Reparameterization

Since the proposed model may include loops, it is necessary to introduce an approximation to calculate mariginal probabilities. For this, we use Tree-based Reparameterization (TRP) (Wainwright et al., 2003) for approximate inference. TRP enumerates a set of spanning trees from the graph. Then, inference is performed by applying an exact inference algorithm such as Belief Propagation to each of the spanning trees, and updates of marginal probabilities are continued until they converge.

4 Experiments

4.1 Dataset

Our dataset is a random selection of 2300 articles from the Japanese version of Wikipedia as of October 2005. All anchor texts appearing under HTML `<LI>` tags are hand-annotated with NE class label. We use the Extended Named Entity Hierarchy (Sekine et al., 2002) as the NE class labeling guideline, but reduce the number of classes to 13 from the original 200+ by ignoring fine-grained categories and nearby categories in order to avoid data sparseness. We eliminate examples that consist of less than two nodes in the SCR model. There are 16136 anchor texts with 14285 NEs. The number of Sibling, Cousin and Relative edges in the dataset are $|E_S| = 4925$, $|E_C| = 13134$ and $|E_R| = 746$ respectively.

4.2 Experimental settings

The aims of experiments are the two-fold. Firstly, we investigate the effect of each cliques. The several graphs are composed with the three sorts of edges. We also compare the graph-based models with a node-wise method – just MaxEnt method not using any edge dependency. Secondly, we compare the proposed method by CRFs with a baseline method by Support Vector Machines (SVMs) (Vapnik, 1998).

The experimental settings of CRFs and SVMs are as follows.

CRFs In order to investigate which type of clique boosts classification performance, we perform experiments on several CRFs models that are constructed from combinations of defined cliques. Re-

	SCR	SC	SR	CR
# of loopy examples	318 (36%)	324 (32%)	101 (1%)	42 (2%)
# of linear chain or tree examples	555 (64%)	631 (62%)	2883 (27%)	1464 (54%)
# of one node examples	0 (0%)	60 (6%)	7800 (72%)	1176 (44%)
# of total examples	873	1015	10784	2682
average # of nodes per example	18.5	15.8	1.5	6.0
	S	C	R	I
# of loopy examples	0 (0%)	0 (0%)	0 (0%)	0 (0%)
# of linear chain or tree examples	2913 (26%)	1631 (54%)	237 (2%)	0 (0%)
# of one node examples	8298 (74%)	1380 (46%)	15153 (98%)	16136 (100%)
# of total examples	11211	3011	15390	16136
average # of nodes per example	1.4	5.4	1.05	1

Table 1: The dataset details constructed from each model.

sulting models of CRFs evaluated on this experiments are SCR, SC, SR, CR, S, C, R and I (independent). Figure 3 shows representative graphs of the eight models. When the graph are disconnected by reducing the edges, the classification is performed on each connected subgraph. We call it an *example*. We name the *examples* according the graph structure: ”loopy examples” are subgraphs including at least one cycle; ”linear chain or tree examples” are subgraphs including not a cycle but at least an edge; ”one node examples” are subgraphs without edges. Table 1 shows the distribution of the examples of each model. Since SCR, SC, SR and CR model have loopy examples, TRP approximate inference is necessary. To perform training and testing with CRFs, we use GRMM (Sutton, 2006) with TRP. We set the Gaussian Prior variances for weights as $\sigma^2 = 10$ in all models.

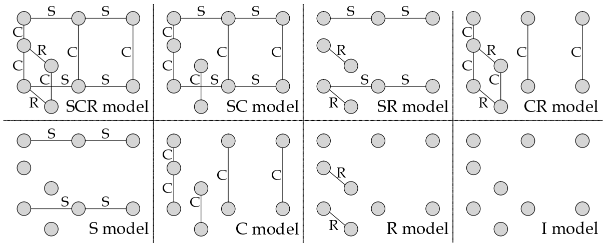


Figure 3: An example of graphs constructed by combination of defined cliques. S, C, R in the model names mean that corresponding model has Sibling, Cousin, Relative cliques respectively. In each model, classification is performed on each connected subgraph.

SVMs We introduce two models by SVMs (model I and model P). In model I, each anchor text is classified independently. In model P, we ordered the anchor texts in a linear-chain sequence. Then, we perform a history-based classification along the sequence, in which $j-1$-th classification result is used in j-th classification. We use TinySVM with a linear-kernel. One-versus-rest method is used for multi-class classification. To perform training and testing with SVMs, we use TinySVM [2] with a linear-kernel, and one-versus-rest is used for multi-class classification. We used the cost of constraint violation $C = 1$.

Features for CRFs and SVMs The features used in the classification with CRFs and SVMs are shown in Table 2. Japanese morphological analyzer MeCab [3] is used to obtain morphemes.

4.3 Evaluation

We evaluate the models by 5 fold cross-validation. Since the number of examples are different in each model, the datasets are divided taking the examples – namely, connected subgraphs – in SCR model. The size of divided five sub-data are roughly equal. We evaluate per-class and total extraction performance by F1-value.

4.4 Results and discussion

Table 3 shows the classification accuracy of each model. The second column “N” stands for the number of nodes in the gold data. The second last row “ALL” stands for the F1-value of all NE classes.

[2] http://www.chasen.org/~taku/software/TinySVM/

[3] http://mecab.sourceforge.net/

types	feature	SVMs	CRFs
observation features	definition (bag-of-words)	√	√ (V)
	heading of articles	√	√ (V)
	heading of articles (morphemes)	√	√ (V)
	categories articles	√	√ (V)
	categories articles (morphemes)	√	√ (V)
	anchor texts	√	√ (V)
	anchor texts (morphemes)	√	√ (V)
	parent tags of anchor texts	√	√ (V)
	text included in the last header of anchor texts	√	√ (V)
	text included in the last header of anchor texts(morphemes)	√	√ (V)
label features	between-label feature		√ (S, C, R)
	previous label	√	

Table 2: Features used in experiments. ”√” means that the corresponding features are used in classification. The V, S, C and R in CRFs column corresponds to the node, sibling edges, cousin edges and relative edges respectively.

		CRFs								SVMs	
NE CLASS	N	C	CR	I	R	S	SC	SCR	SR	I	P
PERSON	3315	.7419	.7429	.7453	.7458	.7507	.7533	**.7981**	.7515	.7383	.7386
TIMEX/NUMEX	2749	.9936	**.9944**	.9940	.9936	.9938	.9931	.9933	.9940	.9933	.9935
FACILITY	2449	.8546	.8541	.8540	.8516	.8500	.8530	.8495	.8495	.8504	**.8560**
PRODUCT	1664	.7414	**.7540**	.7164	.7208	.7130	.7371	.7418	.7187	.7154	.7135
LOCATION	1480	**.7265**	.7239	.6989	.7048	.6974	.7210	.7232	.7033	.7022	.7132
NATURAL_OBJECTS	1132	.3333	.3422	.3476	.3513	.3547	.3294	.3304	.3316	**.3670**	.3326
ORGANIZATION	991	.7122	.7160	.7100	.7073	.7122	.6961	.5580	.7109	.7141	**.7180**
VOCATION	303	.9088	.9050	.9075	.9059	.9150	.9122	.9100	**.9186**	.9091	.9069
EVENT	121	.2740	.2345	.2533	.2667	.2800	.2740	.2759	.2667	.3418	**.3500**
TITLE	42	.1702	.0889	.2800	.2800	**.3462**	.2083	.1277	**.3462**	.2593	.2642
NAME_OTHER	24	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	**.0690**	.0000
UNIT	15	.2353	.1250	.2353	.2353	.2353	.1250	.1250	.2353	**.3333**	.3158
ALL	14285	.7846	**.7862**	.7806	.7814	.7817	.7856	.7854	.7823	.7790	.7798
ALL (no articles)	3898	.5476	**.5495**	.5249	.5274	.5272	.5484	.5465	.5224	.5278	.5386

Table 3: Comparison of F1-values of CRFs and SVMs.

The last row "ALL (no article)" stands for the F1-value of all NE classes which have no gloss texts in Wikipedia.

Relational vs. Independent Among the models constructed by combination of defined cliques, the best F1-value is achieved by CR model, followed by SC, SCR, C, SR, S, R and I. We performed McNemar paired test on labeling disagreements between CR model of CRFs and I model of CRFs. The difference was significant ($p < 0.01$). These results show that considering dependencies work positively in obtaining better accuracy than classifying independently. The Cousin cliques provide the highest accuracy improvement among the three defined cliques. The reason may be that the Cousin cliques appear frequently in comparison with the other cliques, and also possess strong dependencies among anchor texts. As for PERSON, better accuracy is achieved in SC and SCR models. In fact, the PERSON-PERSON pairs frequently appear in Sibling cliques (435 out of 4925) and in Cousin cliques (2557 out of 13125) in the dataset. Also, as for PRODUCT and LOCATION, better accuracy is achieved in the models that contain Cousin cliques (C, CR, SC and SCR model). 1072 PRODUCT-PRODUCT pairs and 738 LOCATION-LOCATION pairs appear in Cousin cliques. "All (no article)" row in Table 3 shows the F1-value of nodes which have no gloss texts. The F1-value difference between CR and I model of CRF in "ALL (no article)" row is larger than the difference in "All" row. The fact means that the dependency information helps to extract NEs without gloss texts in Wikipedia. We attempted a different parameter tying in which the SCR potential functions are tied with a particular observation feature. This parameter tying is introduced by Ghamrawi and McCallum (2005). However, we did not get any improved accuracy.

CRFs vs. SVMs The best model of CRFs (CR model) outperforms the best model of SVMs (P model). We performed McNemar paired test on labeling disagreements between CR model of CRFs and P model of SVMs. The difference was significant ($p < 0.01$). In the classes having larger number of examples, models of CRFs achieve better F1-values than models of SVMs. However, in several classes having smaller number of examples such as EVENT and UNIT, models of SVMs achieve significantly better F1-values than models of CRFs.

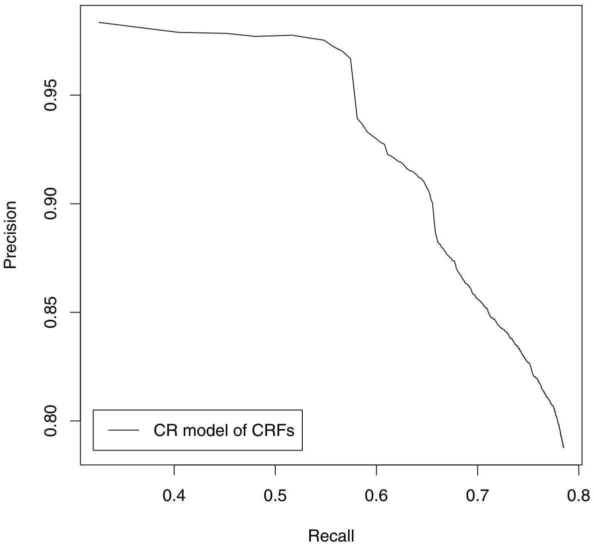


Figure 4: Precision-Recall curve obtained by varying the threshold τ of marginal probability from 1.0 to 0.0.

Filtering NE Candidates using Marginal Probability The precision-recall curve obtained by thresholding the marginal probability of the MAP estimation in the CR models is shown in Figure 4. The curve reaches a peak at 0.57 in recall, and the precision value at that point is 0.97. This precision and recall values mean that 57% of all NEs can be classified with approximately 97% accuracy on a particular thresholding of marginal probability. This results suggest that the extracted NE candidates can be filtered with fewer cost by exploiting the marginal probability.

Training Time The total training times of all CRFs and SVMs models are shown in Table 4. The training time tends to increase in case models have complicated graph structure. For instance, model SCR has complex graph structure compare to model I, therefore the SCR's training time is three times longer than model I. Training the models by SVMs are faster than training the models by CRFs. The difference comes from the implementation issues: C++

	CRFs								SVMs	
	C	CR	I	R	S	SC	SCR	SR	I	P
Training Time (minutes)	207	255	97	90	138	305	316	157	28	29

Table 4: Training Time (minutes)

vs. Java, differences of feature extraction modules, and so on. So, the comparing these two is not the important issue in this experiment.

5 Related Work

Wikipedia has become a popular resource for NLP. Bunescu and Pasca used Wikipedia for detecting and disambiguating NEs in open domain texts (2006). Strube and Ponzetto explored the use of Wikipedia for measuring Semantic Relatedness between two concepts (2006), and for Coreference Resolution (2006).

Several CRFs have been explored for information extraction from the web. Tang et al. proposed Tree-structured Conditional Random Fields (TCRFs) (2006) that capture hierarchical structure of web documents. Zhu et al. proposed Hierarchical Conditional Random Fields (HCRFs) (2006) for product information extraction from Web documents. TCRFs and HCRFs are similar to our approach described in section 4 in that the model structure is induced by page structure. However, the model structures of these models are different from our model.

There are statistical models that capture dependencies between examples. There are two types of classification approaches: iterative (Lu and Getoor, 2003b; Lu and Getoor, 2003a) or collective (Getoor et al., 2001; Taskar et al., 2002). Lu et al. (2003a; 2003b) proposed link-based classification method based on logistic regression. This model iterates local classification until label assignments converge. The results vary from the ordering strategy of local classification. In contrast to iterative classification methods, collective classification methods directly estimate most likely assignments. Getoor et al. proposed Probabilistic Relational Models (PRMs) (2001) which are built upon Bayesian Networks. Since Bayesian Networks are directed graphical models, PRMs cannot model directly the cases where instantiated graph contains cycles. Taskar et al. proposed Relational Markov Networks (RMNs) (2002). RMNs are the special case of Conditional Markov Networks (or Conditional Random Fields) in which graph structure and parameter tying are determined by SQL-like form.

As for the marginal probability to use as a confidence measure shown in Figure 4, Peng et al. (2004) has applied linear-chain CRFs to Chinese word segmentation. It is calculated by constrained forward-backward algorithm (Culotta and McCallum, 2004), and confident segments are added to the dictionary in order to improve segmentation accuracy.

6 Conclusion

In this paper, we proposed a method for categorizing NEs in Wikipedia. We defined three types of cliques that are constitute dependent anchor texts in construct CRFs graph structure, and introduced potential functions for them to reflect classification. The experimental results show that the effectiveness of capturing dependencies, and proposed CRFs model can achieve significant improvements compare to baseline methods with SVMs. The results also show that the dependency information from the HTML tree helps to categorize entities without gloss texts in Wikipedia. The marginal probability of MAP assignments can be used as confidence measure of the entity categorization. We can control the precision by filtering the confidence measure as PR curve in Figure 4. The measure can be also used as a confidence estimator in active learning in CRFs (Kim et al., 2006), where examples with the most uncertainty are selected for presentation to human annotators.

In future research, we plan to explore NE categorization with more fine-grained label set. For NLP applications such as QA, NE dictionary with fine-grained label sets will be a useful resource. However, generally, classification with statistical methods becomes difficult in case that the label set is large, because of the insufficient positive examples. It is an issue to be resolved in the future.

References

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL).*

Lise Getoor, Eran Segal, Ben Taskar, and Daphne Koller. 2001. Probabilistic models of text and link structure for hypertext classification. In *IJCAI Workshop on Text Learning: Beyond Supervision, 2001.*

Nadia Ghamrawi and Andrew McCallum. 2005. Collective multi-label classification. In *Fourteenth Conference on Information and Knowledge Management (CIKM).*

Juanzi Li Jie Tang, Mingcai Hong and Bangyong Liang. 2006. Tree-structured conditional random fields for semantic annotation. In *Proceedings of 5th International Conference of Semantic Web (ISWC-06).*

Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. 2006. MMR-based active machine learning for bio named entity recognition. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL06).*

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Dong C. Liu and Jorge Nocedal. 1989. The limited memory BFGS methods for large scale optimization. In *Mathematical Programming 45*.

Qing Lu and Lise Getoor. 2003a. Link-based classification using labeled and unlabeled data. In *Proceedings of the International Conference On Machine Learning*, Washington DC, August.

Qing Lu and Lise Getoor. 2003b. Link-based text classification. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Andrew McCallum, Khashayar Rohanimanesh, and Charles Sutton. 2003. Dynamic conditional random fields for jointly labeling multiple sequences. In *NIPS Workshop on Syntax, Semantics, and Statistics*, December.

J. Neville and D. Jensen. 2000. Iterative classification in relational data. In *Proceedings of AAAI-2000 Workshop on Learning Statistical Models from Relational Data, pages 13–20. AAAI Press.*

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of The 20th International Conference on Computational Linguistics (COLING).*

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. 2002. Extended named entity hierarchy. In *Proceedings of the LREC-2002.*

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06).*

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the 21th International Conference on Machine Learning.*

Charles Sutton. 2006. GRMM: A graphical models toolkit. http://mallet.cs.umass.edu.

Ben Taskar, Pieter Abbeel, and Daphne Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Vladimir Vapnik. 1998. *Statistical Learning Theory*. Wiley Interscience.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. 2003. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 45(9):1120–1146.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2006. Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

MavenRank: Identifying Influential Members of the US Senate Using Lexical Centrality

Anthony Fader
University of Michigan
afader@umich.edu

Dragomir Radev
University of Michigan
radev@umich.edu

Michael H. Crespin
The University of Georgia
crespin@uga.edu

Burt L. Monroe
The Pennsylvania State University
burtmonroe@psu.edu

Kevin M. Quinn
Harvard University
kevin_quinn@harvard.edu

Michael Colaresi
Michigan State University
colaresi@msu.edu

Abstract

We introduce a technique for identifying the most salient participants in a discussion. Our method, MavenRank is based on lexical centrality: a random walk is performed on a graph in which each node is a participant in the discussion and an edge links two participants who use similar rhetoric. As a test, we used MavenRank to identify the most influential members of the US Senate using data from the *US Congressional Record* and used committee ranking to evaluate the output. Our results show that MavenRank scores are largely driven by committee status in most topics, but can capture speaker centrality in topics where speeches are used to indicate ideological position instead of influence legislation.

1 Introduction

In a conversation or debate between a group of people, we can think of two remarks as interacting if they are both comments on the same topic. For example, if one speaker says "taxes should be lowered to help business," while another argues "taxes should be raised to support our schools," the speeches are interacting with each other by describing the same issue. In a debate with many people arguing about many different things, we could imagine a large network of speeches interacting with each other in the same way. If we associate each speech in the network with its speaker, we can try to identify the most important people in the debate based on how central their speeches are in the network.

To describe this type of centrality, we borrow a term from *The Tipping Point* (Gladwell, 2002), in which Gladwell describes a certain type of personality in a social network called a *maven*. A maven is a trusted expert in a specific field who influences other people by passing information and advice. In this paper, our goal is to identify authoritative speakers who control the spread of ideas within a topic. To do this, we introduce MavenRank, which measures the centrality of speeches as nodes in the type of network described in the previous paragraph.

Significant research has been done in the area of identifying central nodes in a network. Various methods exist for measuring centrality, including degree centrality, closeness, betweenness (Freeman, 1977; Newman, 2003), and eigenvector centrality. Eigenvector centrality in particular has been successfully applied to many different types of networks, including hyperlinked web pages (Brin and Page, 1998; Kleinberg, 1998), lexical networks (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Kurland and Lee, 2005; Kurland and Lee, 2006), and semantic networks (Mihalcea et al., 2004). The authors of (Lin and Kan, 2007) extended these methods to include timestamped graphs where nodes are added over time and applied it to multi-document summarization. In (Tong and Faloutsos, 2006), the authors use random walks on a graph as a method for finding a subgraph that best connects some or all of a set of query nodes. In our paper, we introduce a new application of eigenvector centrality for identifying the central speakers in the type of debate or conversation network described above. Our method is based on the one described in (Erkan

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 658–666, Prague, June 2007. ©2007 Association for Computational Linguistics

and Radev, 2004) and (Mihalcea and Tarau, 2004), but modified to rank *speakers* instead of documents or sentences.

In our paper, we apply our method to analyze the *US Congressional Record*, which is a verbatim transcript of speeches given in the United States House of Representatives and Senate. The *Record* is a dense corpus of speeches made by a large number of people over a long period of time. Using the transcripts of political speeches adds an extra layer of meaning onto the measure of speaker centrality. The centrality of speakers in Congress can be thought of as a measure of relative importance or influence in the US legislative process. We can also use speaker centrality to analyze committee membership: are the central speakers on a given issue ranking members of a related committee? Is there a type of importance captured through speaker centrality that isn't obvious in the natural committee rankings?

There has been growing interest in using techniques from natural language processing in the area of political science. In (Porter et al., 2005) the authors performed a network analysis of members and committees of the US House of Representatives. They found connections between certain committees and political positions that suggest that committee membership is not determined at random. In (Thomas et al., 2006), the authors use the transcripts of debates from the US Congress to automatically classify speeches as supporting or opposing a given topic by taking advantage of the voting records of the speakers. In (Wang et al., 2005), the authors use a generative model to simultaneously discover groups of voters and topics using the voting records and the text from bills of the US Senate and the United Nations. The authors of (Quinn et al., 2006) introduce a multinomial mixture model to perform unsupervised clustering of Congressional speech documents into topically related categories. We rely on the output of this model to cluster the speeches from the *Record* in order to compare speaker rankings within a topic to related committees.

We take advantage of the natural measures of prestige in Senate committees and use them as a standard for comparison with MavenRank. Our hypothesis is that MavenRank centrality will capture the importance of speakers based on the natural committee rankings and seniority. We can test this claim by clustering speeches into topics and then mapping the topics to related committees. If the hypothesis is correct, then the speaker centrality should be correlated with the natural committee rankings.

There have been other attempts to link floor participation with topics in political science. In (Hall, 1996), the author found that serving on a committee can positively predict participation in Congress, but that seniority was not a good predictor. His measure only looked at six bills in three committees, so his method is by far not as comprehensive as the one that we present here. Our approach with MavenRank differs from previous work by providing a large scale analysis of speaker centrality and bringing natural language processing techniques to the realm of political science.

2 Data

2.1 The US Congressional Speech Corpus

The text used in the experiments is from the United States Congressional Speech corpus (Monroe et al., 2006), which is an XML formatted version of the electronic *United States Congressional Record* from the Library of Congress[1]. The *Congressional Record* is a verbatim transcript of the speeches made in the US House of Representatives and Senate beginning with the 101st Congress in 1998 and includes tens of thousands of speeches per year. In our experiments we focused on the records from the 105th and 106th Senates. The basic unit of the US Congressional Speech corpus is a *record*, which corresponds to a single subsection of the print version of the *Congressional Record* and may contain zero or more speakers. Each paragraph of text within a record is tagged as either speech or non-speech and each paragraph of speech text is tagged with the unique id of the speaker. Figure 1 shows an example record file for the sixth record on July 14th, 1997 in the 105th Senate.

In our experiments we use a smaller unit of analysis called a *speech document* by taking all of the text of a speaker within a single record. The capitalization and punctuation is then removed from the text as in (Monroe et al., 2006) and then the

[1] http://thomas.loc.gov

text stemmed using Porter's Snowball II stemmer[2]. Figure 1 shows an example speech document for speaker 15703 (Herb Kohl of Wisconsin) that has been generated from the record in Figure 1.

In addition to speech documents, we also use *speaker documents*. A speaker document is the concatenation of all of a speaker's speech documents within a single session and topic (so a single speaker may have multiple speaker documents across topics). For example within the 105th Senate in topic 1 ("Judicial Nominations"), Senator Kohl has four speech documents, so the speaker document attributed to him within this session and topic would be the text of these four documents treated as a single unit. The order of the concatenation does not matter since we will look at it as a vector of weighted term frequencies (see Section 3.2).

2.2 Topic Clusters

We used the direct output of the 42-topic model of the 105th-108th Senates from (Quinn et al., 2006) to further divide the speech documents into topic clusters. In their paper, they use a model where the probabilities of a document belonging to a certain topic varies smoothly over time and the words within a given document have exactly the same probability of being drawn from a particular topic. These two properties make the model different than standard mixture models (McLachlan and Peel, 2000) and the latent Dirichlet allocation model of (Blei et al., 2003). The model of (Quinn et al., 2006) is most closely related to the model of (Blei and Lafferty, 2006), who present a generalization of the model used by (Quinn et al., 2006). Table 1 lists the 42 topics and their related committees.

The output from the topic model is a $D \times 42$ matrix $\mathbf{Z}$ where D is the number of speech documents and the element z_{dk} represents the probability of the dth speech document being generated by topic k. We clustered the speech documents by assigning a speech document d to the kth cluster where

$$k = \arg\max_{j} z_{dj}.$$

If the maximum value is not unique, we arbitrarily assign d to the lowest numbered cluster where z_{dj} is a maximum. A typical topic cluster contains several hundred speech documents, while some of the larger topic clusters contain several thousand.

2.3 Committee Membership Information

The committee membership information that we used in the experiments is from Stewart and Woon's committee assignment codebook (Stewart and Woon, 2005). This provided us with a roster for each committee and rank and seniority information for each member. In our experiments we use the *rank within party* and *committee seniority* member attributes to test the output of our pipeline. The rank within party attribute orders the members of a committee based on the Resolution that appointed the members with the highest ranking members having the lowest number. The chair and ranking members always receive a rank of 1 within their party. A committee member's committee seniority attribute corresponds to the number of years that the member has served on the given committee.

2.4 Mapping Topics to Committees

In order to test our hypothesis that lexical centrality is correlated with the natural committee rankings, we needed a map from topics to related committees. We based our mapping on Senate Rule XXV,[3] which defines the committees, and the descriptions on committee home pages. Table 1 shows the map, where a topic's related committees are listed in italics below the topic name. Because we are matching short topic names to the complex descriptions given by Rule XXV, the topic-committee map is not one to one or even particularly well defined: some topics are mapped to multiple committees, some topics are not mapped to any committees, and two different topics may be mapped to the same committee. This is not a major problem because even if a one to one map between topics and committees existed, speakers from outside a topic's related committee are free to participate in the topic simply by giving a speech. Therefore there is no way to rank all speakers in a topic using committee information. To test our hypotheses, we focused our attention on topics that have at least one related committee. In Section 4.3 we describe how the MavenRank scores

[2] http://snowball.tartarus.org/algorithms/english/stemmer.html

[3] http://rules.senate.gov/senaterules/rule25.php

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE RECORD SYSTEM "record.dtd">
<RECORD>
  <HEADER>
    <CHAMBER>Senate</CHAMBER>
    <TITLE>NOMINATION OF JOEL KLEIN TO BE ASSISTANT ATTORNEY
    GENERAL IN CHARGE OF THE ANTITRUST DIVISION </TITLE>
    <DATE>19970714</DATE>
  </HEADER>
  <BODY>
    <GRAF>
      <PAGEREF></PAGEREF>
      <SPEAKER>NULL</SPEAKER>
      <NONSPEECH>NOMINATION OF JOEL KLEIN TO BE ASSISTANT
      ATTORNEY GENERAL IN CHARGE OF THE ANTITRUST DIVISION
      (Senate - July 14, 1997)</NONSPEECH>
    </GRAF>
    <GRAF>
      <PAGEREF>S7413</PAGEREF>
      <SPEAKER>15703</SPEAKER>
      <SPEECH> Mr. President, as the ranking Democrat on the
      Antitrust Subcommittee, let me tell you why I support Mr.
      Klein's nomination, why he is a good choice for the job,
      and why we ought to confirm him today.
      </SPEECH>
    </GRAF>
    . . .
    <GRAF>
      <PAGEREF>S7414</PAGEREF>
      <SPEAKER>UNK1</SPEAKER>
      <SPEECH> Without objection, it is so ordered.  </SPEECH>
    </GRAF>
  </BODY>
</RECORD>
```

```
mr presid a the rank democrat on the antitrust subcommitte
let me tell you why i support mr klein nomin why he i a
good choic for the job and why we ought to confirm him
todai
first joel klein i an accomplish lawyer with a distinguish
career he graduat from columbia univers and harvard law
school and clerk for the u court of appeal here in
washington then for justic powel just a importantli he i
the presid choic to head the antitrust divis and i believ
that ani presid democrat or republican i entitl to a strong
presumpt in favor of hi execut branch nomine second joel
klein i a pragmatist not an idealogu hi answer at hi confirm
hear suggest that he i not antibusi a some would claim the
antitrust divis wa in the late 1970 nor anticonsum a some
argu the divis wa dure the 1980 instead he will plot a middl
cours i believ that promot free market fair competit and
consum welfar
the third reason we should confirm joel klein i becaus no on
deserv to linger in thi type of legisl limbo here in congress
we need the input of a confirm head of the antitrust divis
to give u the administr view on a varieti of import polici
matter defens consolid electr deregul and telecommun merger
among other we need someon who can speak with author for the
divis without a cloud hang over hi head
more than that without a confirm leader moral at the
antitrust divis i suffer and given the pace at which the
presid ha nomin and the senat ha confirm appointe if we fail
to approv mr klein it will be at least a year befor we confirm
a replac mayb longer and mayb never so we need to act now we
can't afford to let the antitrust divis continu to drift
final mr presid i have great respect for the senat from south
carolina a well a the senat from nebraska and north dakota
thei have been forc advoc for consum on telecommun matter and
. . .
```

Figure 1: A sample of the text from record 105.sen.19970714.006.xml and the speech document for Senator Herb Kohl of Wisconsin (id 15703) generated from it. The "…" represents omitted text.

1	Judicial Nominations	*Judiciary*
2	Law & Crime 1 (Violence / Drugs)	*Judiciary*
3	Banking / Finance	*Banking, Housing, and Urban Affairs*
4	Armed Forces 1 (Manpower)	*Armed Services*
5	Armed Forces 2 (Infrastructure)	*Armed Services*
6	Symbolic (Tribute - Living)	
7	Symbolic (Congratulations - Sports)	
8	Energy	*Energy and Natural Resources*
9	Defense (Use of Force)	*Armed Services* *Homeland Security and Governmental Affairs* *Intelligence (Select Committee)*
10	Jesse Helms re Debt	
11	Environment 1 (Public Lands)	*Energy and Natural Resources* *Agriculture, Nutrition, and Forestry*
12	Health 1 (Medical)	*Health, Education, Labor, and Pensions*
13	International Affairs (Arms Control)	*Foreign Relations*
14	Social Welfare	*Agriculture, Nutrition, and Forestry* *Banking, Housing, and Urban Affairs* *Health, Education, Labor, and Pensions* *Finance*
15	Health 2 (Economics - Seniors)	*Health, Education, Labor, and Pensions* *Veterans' Affairs* *Agriculture, Nutrition, and Forestry* *Aging (Special Committee)* *Finance*
16	Gordon Smith re Hate Crime	
17	Debt / Deficit / Social Security	*Appropriations* *Budget* *Finance* *Aging (Special Committee)*
18	Supreme Court / Constitutional	*Judiciary*
19	Commercial Infrastructure	*Commerce, Science, and Transportation*
20	Symbolic (Remembrance - Military)	
21	International Affairs (Diplomacy)	*Foreign Relations*
22	Abortion	*Judiciary* *Health, Education, Labor, and Pensions*
23	Symbolic (Tribute - Constituent)	
24	Agriculture	*Agriculture, Nutrition, and Forestry*
25	Intelligence	*Intelligence (Select Committee)* *Homeland Security and Governmental Affairs*
26	Health 3 (Economics - General)	*Health, Education, Labor, and Pensions* *Finance*
27	Procedural 1 (Housekeeping 1)	
28	Procedural 2 (Housekeeping 2)	
29	Campaign Finance	*Rules and Administration*
30	Law & Crime 2 (Federal)	*Judiciary*
31	Child Protection	*Health, Education, Labor, and Pensions* *Agriculture, Nutrition, and Forestry*
32	Labor 1 (Workers, esp. Retirement)	*Health, Education, Labor, and Pensions* *Aging (Special Committee)* *Small Business and Entrepreneurship*
33	Environment 2 (Regulation)	*Environment and Public Works* *Agriculture, Nutrition, and Forestry* *Energy and Natural Resources*
34	Procedural 3 (Legislation 1)	
35	Procedural 4 (Legislation 2)	
36	Procedural 5 (Housekeeping 3)	
37	Procedural 6 (Housekeeping 4)	
38	Taxes	*Finance*
39	Symbolic (Remembrance - Nonmilitary)	
40	Labor 2 (Employment)	*Health, Education, Labor, and Pensions* *Small Business and Entrepreneurship*
41	Foreign Trade	*Finance* *Banking, Housing, and Urban Affairs*
42	Education	*Health, Education, Labor, and Pensions*

Table 1: The numbers and names of the 42 topics from (Quinn et al., 2006) with our mappings to related committees (listed below the topic name, if available).

of speakers who are not members of related committees were taken into account when we measured the rank correlations.

3 MavenRank and Lexical Similarity

The following sections describe MavenRank, a measure of speaker centrality, and tf-idf cosine similarity, which is used to measure the lexical similarity of speeches.

3.1 MavenRank

MavenRank is a graph-based method for finding speaker centrality. It is similar to the methods in (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Kurland and Lee, 2005), which can be used for ranking sentences in extractive summaries and documents in an information retrieval system. Given a collection of speeches $s_1, \ldots, s_N$ and a measure of lexical similarity between pairs $\mathrm{sim}(s_i, s_j) \geq 0$, a similarity graph can be constructed. The nodes of the graph represent the speeches and a weighted similarity edge is placed between pairs that exceed a similarity threshold s_{min}. MavenRank is based on the premise that important speakers will have central speeches in the graph, and that central speeches should be similar to other central speeches. A recursive explanation of this concept is that the score of a speech should be proportional to the scores of its similar neighbors.

Given a speech s in the graph, we can express the recursive definition of its score $p(s)$ as

$$p(s) = \sum_{t \in adj[s]} \frac{p(t)}{wdeg(t)} \tag{1}$$

where $adj[s]$ is the set of all speeches adjacent to s and $wdeg(t) = \sum_{u \in adj[t]} \mathrm{sim}(t, u)$, the weighted degree of t. Equation (1) captures the idea that the MavenRank score of a speech is distributed to its neighbors. We can rewrite this using matrix notation as

$$\mathbf{p} = \mathbf{pB} \tag{2}$$

where $\mathbf{p} = (p(s_1), p(s_2), \ldots, p(s_N))$ and the matrix $\mathbf{B}$ is the row normalized similarity matrix of the graph

$$\mathbf{B}(i, j) = \frac{\mathbf{S}(i, j)}{\sum_k \mathbf{S}(i, k)} \tag{3}$$

where $\mathbf{S}(i, j) = \mathrm{sim}(s_i, s_j)$. Equation (2) shows that the vector of MavenRank scores $\mathbf{p}$ is the left eigenvector of $\mathbf{B}$ with eigenvalue 1.

We can prove that the eigenvector $\mathbf{p}$ exists by using a techinque from (Page et al., 1999). We can treat the matrix $\mathbf{B}$ as a Markov chain describing the transition probabilities of a random walk on the speech similarity graph. The vector $\mathbf{p}$ then represents the stationary distribution of the random walk. It is possible that some parts of the graph are disconnected or that the walk gets trapped in a component. These problems are solved by reserving a small escape probability at each node that represents a chance of jumping to any node in the graph, making the Markov chain irreducible and aperiodic, which guarantees the existence of the eigenvector. Assuming a uniform escape probability for each node on the graph, we can rewrite Equation (2) as

$$\mathbf{p} = \mathbf{p}[d\mathbf{U} + (1 - d)\mathbf{B}] \tag{4}$$

where $\mathbf{U}$ is a square matrix with $\mathbf{U}(i, j) = 1/N$ for all i and j, N is the number of nodes, and d is the escape probability chosen in the interval $[0.1, 0.2]$ (Brin and Page, 1998). Equation (4) is known as *PageRank* (Page et al., 1999) and is used for determining prestige on the web in the Google search engine.

3.2 Lexical Similarity

In our experiments, we used tf-idf cosine similarity to measure lexical similarity between speech documents. We represent each speech document as a vector of term frequencies (or *tf*), which are weighted according to the relative importance of the given term in the cluster. The terms are weighted by their *inverse document frequency* or *idf*. The idf of a term w is given by (Sparck-Jones, 1972)

$$\mathrm{idf}(w) = \log\left(\frac{N}{n_w}\right) \tag{5}$$

where N is the number of documents in the corpus and n_w is the number of documents in the corpus containing the term w. It follows that very common words like “of” or “the” have a very low idf, while the idf values of rare words are higher. In our experiments, we calculated the idf values for each topic using all speech documents across sessions within the

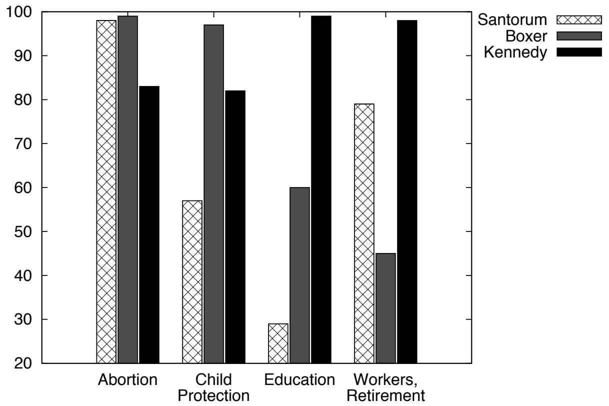


Figure 2: MavenRank percentiles for three speakers over four topics.

given topic. We calculated *topic-specific* idf values because some words may be relatively unimportant in one topic, but important in another. For example, in topic 22 ("Abortion"), the idf of the term "abort" is near 0.20, while in topic 38 ("Taxes"), its idf is near 7.18.

The tf-idf cosine similarity measure $\text{tf-idf-cosine}(u, v)$ is defined as

$$\frac{\sum_{w\in u,v} \mathrm{tf}_u(w)\,\mathrm{tf}_v(w)\,\mathrm{idf}(w)^2}{\sqrt{\sum_{w\in u}(\mathrm{tf}_u(w)\,\mathrm{idf}(w))^2}\sqrt{\sum_{w\in v}(\mathrm{tf}_v(w)\,\mathrm{idf}(w))^2}}, \qquad (6)$$

which is the cosine of the angle between the tf-idf vectors.

There are other alternatives to tf-idf cosine similarity. Some other possible similarity measures are document edit distance, the language models from (Kurland and Lee, 2005), or generation probabilities from (Erkan, 2006). For simplicity, we only used tf-idf similarities in our experiments, but any of these measures could be used in this case.

4 Experiments and Results

4.1 Data

We used the topic clusters from the 105th Senate as training data to adjust the parameter s_{min} and observe trends in the data. We did not run experiments to test the effect of different values of s_{min} on MavenRank scores, but our chosen value of 0.25 has shown to give acceptable results in similar experiments (Erkan and Radev, 2004). We used the topic clusters from the 106th Senate as test data. For the speech document networks, there was an average of 351 nodes (speech documents) and 2142 edges per topic. For the speaker document networks, there was an average of 63 nodes (speakers) and 545 edges per topic.

4.2 Experimental Setup

We set up a pipeline using a Perl implementation of tf-idf cosine similarity and MavenRank. We ran MavenRank on the topic clusters and ranked the speakers based on the output. We used two different types granularities of the graphs as input: one where the nodes are speech documents and another where the nodes are speaker documents (see Section 2.1). For the speech document graph, a speaker's score is determined by the sum of the MavenRank scores of the speeches given by that speaker.

4.3 Evaluation Methods

To evaluate our output, we estimate independent ordinary least squares linear regression models of MavenRank centrality for topics with at least one related committee (there are 29 total):

$$\begin{aligned} MavenRank_{ik} &= \beta_{0k} + \beta_{sk}Seniority_{ik} + \\ &+ \beta_{rk}RankingMember_{jk} + \epsilon_{ik} \qquad (7) \end{aligned}$$

where i indexes Senators, k indexes topics, $Seniority_{ik}$ is the number of years Senator i has served on the relevant committee for topic k (value zero for those not on a relevant committee) and $RankingMember_{jk}$ has the value of one only for the Chair and ranking minority member of a relevant committee. We are interested primarily in the overall significance of the estimated model (indicating committee effects) and, secondarily, in the specific source of any committee effect in seniority or committee rank.

4.4 Results

Table 2 summarizes the results. "Maven" status on most topics does appear to be driven by committee status, as expected. There are particularly strong effects of seniority and rank in topics tied to the Judiciary, Foreign Relations, and Armed Services committees, as well as legislation-rich areas of domestic policy. Perhaps of greater interest are the topics that do not have committee effects. These are of three distinct types. The first are highly politicized topics for which speeches are intended not to influence

	Topic	$p(F)$[a]	$p(\beta_s > 0)$[b]	$p(\beta_r > 0)$[c]
Seniority and Ranking Status Both Significant				
2	Law & Crime 1 [Violent]	$< .001$	0.016	$< .001$
18	Constitutional	$< .001$	0.003	$< .001$
Seniority Significant				
12	Health 1 [Medical]	$< .001$	$< .001$	0.567
42	Education	$< .001$	$< .001$	0.337
41	Trade	$< .001$	$< .001$	0.087
21	Int'l Affairs [Nonmilitary]	$< .001$	0.007	0.338
9	Defense [Use of Force]	0.002	0.001	0.926
19	Commercial Infrastructure	0.007	0.032	0.332
40	Labor 2 [Employment]	0.029	0.010	0.114
38	Taxes	0.037	0.033	0.895
Ranking Status Significant				
30	Crime 2 [Federal]	$< .001$	0.334	$< .001$
8	Energy	$< .001$	0.145	$< .001$
1	Judicial Nominations	$< .001$	0.668	$< .001$
14	Social Welfare	$< .001$	0.072	0.005
13	Int'l Affairs [Arms]	$< .001$	0.759	0.001
4	Armed Forces 1 [Manpower]	0.007	0.180	0.049
Seniority and Ranking Status Jointly Significant				
26	Health 3 [Economics]	0.001	0.106	0.064
32	Labor 1 [Workers]	0.007	0.156	0.181
33	Environment 2 [Regulation]	0.007	0.063	0.056
3	Banking / Finance	0.042	0.141	0.579
No Significant Effects of Committee Status				
11	Environment 1 [Public Lands]	0.104	0.102	0.565
22	Abortion	0.419	0.609	0.252
5	Armed Forces 2 [Infrastructure]	0.479	0.267	0.919
24	Agriculture	0.496	0.643	0.425
17	Debt / Social Security	0.502	0.905	0.295
15	Health 2 [Seniors]	0.706	0.502	0.922
25	Intelligence	0.735	0.489	0.834
29	Campaign Finance	0.814	0.748	0.560
31	Child Protection	0.856	0.580	0.718

[a] F-test for joint significance of committee variables.
[b] T-test for significance of committee seniority.
[c] T-test for significance of chair or ranking member status.

Table 2: Significance tests for ordinary least squares (OLS) linear regressions of MavenRank scores (Speech-documents graph) on committee seniority (in years) and ranking status (chair or ranking member), 106th Senate, topic-by-topic. Results for the speaker-documents graph are similar.

legislation as much as indicate an ideological or partisan position, so the mavens are not on particular committees (abortion, children, seniors, the economy). The second are "distributive politics" topics where many Senators speak to defend state or regional interests, so debate is broadly distributed and there are no clear mavens (agriculture, military base closures, public lands). Third are topics where there are not enough speeches for clear results, because most debate occurred after 1999-2000 (post-9/11 intelligence reform, McCain-Feingold campaign finance reform).

Alternative models, using measures of centrality based on the centroid were also examined. Distance to centroid provides broadly similar results as MavenRank, with several marginal significance results reversed in each direction. Cosine similarity with centroid, on the other hand, appears to have no relationship with committee structure.

Figure 2 shows the MavenRank percentiles (using the speech document network) for Senators Rick Santorum, Barbara Boxer, and Edward Kennedy across a few topics in the 106th Senate. These sample scores conform to the expected rankings for these speakers. In this session, Santorum was the sponsor of a bill to ban partial birth abortions and was a spokesman for Social Security reform, which support his high ranking in abortion and workers/retirement. Boxer acted as the lead opposition to Santorum's abortion bill and is known for her support of child abuse laws. Kennedy was ranking member of the Health, Education, Labor, and Pensions committee and the Judiciary committee (which was involved with the abortion bill).

4.5 MavenRank in Other Contexts

MavenRank is a general method for finding central speakers in a discussion and can be applied to areas outside of political science. One potential application would be analyzing blog posts to find "Maven" bloggers by treating blogs as speakers and posts as speeches. Similarly, MavenRank could be used to find central participants in a newsgroup, a forum, or a collection of email conversations.

5 Conclusion

We have presented a technique for identifying lexically central speakers using a graph based method called MavenRank. To test our method for finding central speakers, we analyzed the *Congressional*

Record by creating a map from the clusters of speeches to Senate committees and comparing the natural ranking committee members to the output of MavenRank. We found evidence of a possible relationship between the lexical centrality and committee rank of a speaker by ranking the speeches using MavenRank and computing the rank correlation with the natural ordering of speakers. Some specific committees disagreed with our hypothesis that MavenRank and committee position are correlated, which we propose is because of the non-legislative aspects of those specific committees. The results of our experiment suggest that MavenRank can indeed be used to find central speakers in a corpus of speeches.

We are currently working on applying our methods to the US House of Representatives and other records of parliamentary speech from the United Kingdom and Australia. We have also developed a dynamic version of MavenRank that takes time into account when finding lexical centrality and plan on using it with the various parliamentary records. We are interested in dynamic MavenRank to go further with the idea of tracking how ideas get propagated through a network of debates, including congressional records, blogs, and newsgroups.

Acknowledgments

This paper is based upon work supported by the National Science Foundation under Grant No. 0527513, “DHB: The dynamics of Political Representation and Political Rhetoric”. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

David Blei and John Lafferty. 2006. Dynamic topic models. In *Machine Learning: Proceedings of the Twenty-Third International Conference (ICML)*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

Güneş Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*.

Gunes Erkan. 2006. Language model-based document clustering using random walks. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 479–486, New York City, USA, June. Association for Computational Linguistics.

L. C. Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, March.

Malcolm Gladwell. 2002. *The Tipping Point: How Little Things Can Make a Big Difference*. Back Bay Books, January.

Richard L. Hall. 1996. *Participation in Congress*. Yale University Press.

Jon M. Kleinberg. 1998. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677.

Oren Kurland and Lillian Lee. 2005. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of SIGIR*, pages 306–313.

Oren Kurland and Lillian Lee. 2006. Respect my authority! HITS without hyperlinks, utilizing cluster-based language models. In *Proceedings of SIGIR*, pages 83–90.

Ziheng Lin and Min-Yen Kan. 2007. Timestamped graphs: Evolutionary models of text for multi-document summarization. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 25–32, Rochester, NY, USA. Association for Computational Linguistics.

Geoffrey McLachlan and David Peel. 2000. *Finite Mixture Models*. New York: Wiley.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the Ninth Conference on Empirical Methods in Natural Language Processing (EMNLP ’04)*.

Rada Mihalcea, Paul Tarau, and Elizabeth Figa. 2004. Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of the Twentieth International Conference on Computational Linguistics (COLING ’04)*, pages 1126–1132.

Burt L. Monroe, Cheryl L. Monroe, Kevin M. Quinn, Dragomir Radev, Michael H. Crespin, Michael P. Colaresi, Anthony Fader, Jacob Balazer, and Steven P. Abney. 2006. United states congressional speech corpus. Department of Political Science, The Pennsylvania State University.

Mark E. J. Newman. 2003. A measure of betweenness centrality based on random walks. Technical Report cond-mat/0309045, Arxiv.org.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the Web. Technical Report 1999-66, Stanford Digital Library Technologies Project, Stanford University, November 11,.

Mason A. Porter, Peter J. Mucha, M. E. J. Newman, and Casey M. Warmbrand. 2005. A network analysis of committees in the u.s. house of representatives. *PNAS*, 102(20), May.

Kevin M. Quinn, Burt L. Monroe, Michael Colaresi, Michael H. Crespin, and Dragomir R. Radev. 2006. An automated method of topic-coding legislative speech over time with application to the 105th-108th U.S. senate. In *Midwest Political Science Association Meeting*.

K. Sparck-Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–20.

Charles Stewart and Jonathan Woon. 2005. Congressional committee assignments, 103rd to 105th congresses, 1993–1998: Senate, july 12, 2005. `http://web.mit.edu/17.251/www/data_page.html`.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327–335.

Hanghang Tong and Christos Faloutsos. 2006. Center-piece subgraphs: problem definition and fast solutions. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *KDD*, pages 404–413. ACM.

Xuerui Wang, Natasha Mohanty, and Andrew McCallum. 2005. Group and topic discovery from relations and their attributes. In *NIPS*.

Bootstrapping Feature-Rich Dependency Parsers with Entropic Priors

David A. Smith and Jason Eisner
Department of Computer Science
Johns Hopkins University
Balitmore, MD 21218, USA
{dasmith,eisner}@jhu.edu

Abstract

One may need to build a statistical parser for a new language, using only a very small labeled treebank together with raw text. We argue that bootstrapping a parser is most promising when the model uses a rich set of redundant features, as in recent models for scoring dependency parses (McDonald et al., 2005). Drawing on Abney's (2004) analysis of the Yarowsky algorithm, we perform bootstrapping by entropy regularization: we maximize a linear combination of conditional likelihood on labeled data and confidence (negative Rényi entropy) on unlabeled data. In initial experiments, this surpassed EM for training a simple feature-poor generative model, and also improved the performance of a feature-rich, conditionally estimated model where EM could not easily have been applied. For our models and training sets, more peaked measures of confidence, measured by Rényi entropy, outperformed smoother ones. We discuss how our feature set could be extended with cross-lingual or cross-domain features, to incorporate knowledge from parallel or comparable corpora during bootstrapping.

1 Motivation

In this paper, we address the problem of bootstrapping new statistical parsers for new languages, genres, or domains.

Why is this problem important? Many applications of multilingual NLP require parsing in order to extract information, opinions, and answers from text, and to produce improved translations. Yet an adequate labeled training corpus—a large **treebank** of manually constructed parse trees of typical sentences—is rarely available and would be prohibitively expensive to develop.

We show how it is possible to train instead from a small hand-labeled treebank in the target domain, together with a large *unannotated* collection of in-domain sentences. Additional resources such as parsers for *other* domains or languages can be integrated naturally.

Dependency parsing is important as a key component in leading systems for information extraction (Weischedel, 2004)[1] and question answering (Peng et al., 2005). These systems rely on edges or paths in dependency parse trees to define their extraction patterns and classification features. Parsing is also key to the latest advances in machine translation, which translate syntactic phrases (Galley et al., 2006; Marcu et al., 2006; Cowan et al., 2006).

2 Our Approach

Our approach rests on three observations:

- Recent "feature-based" parsing models are an excellent fit for bootstrapping, because the parse is often overdetermined by many redundant features.
- The feature-based framework is flexible enough to incorporate other sources of guidance during training or testing—such as the knowledge contained in a parser for another language or domain.
- Maximizing a combination of likelihood on labeled data and confidence on unlabeled data is a principled approach to bootstrapping.

2.1 Feature-Based Parsing

McDonald et al. (2005) introduced a simple, flexible framework for scoring dependency parses. Each directed edge e in the dependency tree is described with a high-dimensional feature vector $\mathbf{f}(e)$. The edge's score is the dot product $\mathbf{f}(e) \cdot \boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is a learned weight vector. The overall score of a dependency tree is the sum of the scores of all edges in the tree.

[1] Ralph Weischedel (p.c.) reports that this system's performance degrades considerably when only phrase chunking is available rather than full parsing.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 667–677, Prague, June 2007. ©2007 Association for Computational Linguistics

Given an n-word input sentence, the parser begins by scoring each of the $O(n^2)$ possible edges, and then seeks the highest-scoring legal dependency tree formed by any $n-1$ of these edges, using an $O(n^3)$ dynamic programming algorithm (Eisner, 1996) for projective trees. For non-projective parsing, $O(n^3)$, or with some trickery $O(n^2)$, greedy algorithms exist (Chu and Liu, 1965; Edmonds, 1967; Gabow et al., 1986).

The feature function $\mathbf{f}$ may pay attention to many properties of the directed edge e. Of course, features may consider the parent and child words connected by e, and their parts of speech.[2] But some features used by McDonald et al. (2005) also consider the parts of speech of words *adjacent to* the parent and child, or *between* the parent and child, as well as the *number* of words between the parent and child. In general, these features are not available in a generative model such as a PCFG.

Although feature-based models are often trained purely discriminatively, we will see in §2.6 how to train them to model conditional probabilities.

2.2 Feature-Based Parsing and Bootstrapping

The above parsing model is robust, thanks to its many features. On the Penn Treebank WSJ sections 02–21, for example, McDonald's parser extracts 5.5 million feature types from supervised edges *alone*, with about 120 feature tokens firing per edge. The highest-scoring parse tree represents a *consensus* among all features on all prospective edges. Even if a prospective edge has some discouraging features (i.e., with negative or zero weights), it may still have a relatively high score thanks to its other features. Furthermore, even if the edge has a *low* total score, it may still appear in the consensus parse if the alternatives are even worse or are incompatible with other high-scoring edges.

Put another way, the parser is not able to include high-scoring features or edges independently of one another. Selecting a good feature means accepting all other features on that edge. It also means rejecting various other edges, because of the global constraints that a legal parse tree must give each word only one parent and must be free of cycles and, in the projective case, crossings.

Our observation is that this situation is ideal for so-called "bootstrapping," "co-training," or "minimally supervised" learning methods (Yarowsky, 1995; Blum and Mitchell, 1998; Yarowsky and Wicentowski, 2000). Such methods should thrive when the right answer is overdetermined owing to redundant features and/or global constraints.

Concretely, suppose we start by training a supervised parser on only 100 examples, using some regularization method to prevent overfitting to this set. While many features might truly be relevant to the task, only a few appear often enough in this small training set to acquire significantly positive or negative weights.

Even this lightly trained parser may be quite sure of itself on *some* test sentences in a large unannotated corpus, when one parse scores far higher than all others. More generally, the parser may be sure about *part* of a sentence: it may be certain that a particular edge is present (or absent), because that edge tends to be present (or absent) in all high-scoring parses.

Retraining the feature weights $\boldsymbol{\theta}$ on these high-confidence edges can learn about additional features that are correlated with an edge's success or failure. For example, it may now learn strong weights for lexically specific features that were never observed in the supervised training set. The retrained parser may now be able to confidently parse even more of the unannotated examples; so we can iterate the process.

Our hope is that the model identifies new good and bad edges at each step, and does so correctly. The *more* features and global constraints the model has,

- the more power it will have to discriminate among edges even when $\boldsymbol{\theta}$ is *insufficiently* trained. (Some feature weights may be too weak (i.e., too close to zero) because the initial labeled set is small.)
- the more robust it will be against errors even when $\boldsymbol{\theta}$ is *incorrectly* trained. (Some feature weights may be too strong or have the wrong sign, because of overfitting or mistaken parses during bootstrapping.)

[2] Note that since we are not trying to *predict* parts of speech, we treat the output of one or more automatic taggers as yet more inputs to edge feature functions.

In the former case, strong features lend their strength to weak ones. In the latter case, a conflict among strong features weakens the ones that depart from the consensus, or discounts the example sentence if there is no consensus.

Previous work on parser bootstrapping has not been able to exploit this redundancy among features, because it has used PCFG-like models with far fewer features (Steedman et al., 2003).

2.3 Adaptation and Projection via Features

The previous section assumed that we had a small supervised treebank in the target language and domain (plus a large unsupervised corpus). We now consider other, more dubious, knowledge sources that might supplement or replace this small treebank. In each case, we can use these knowledge sources to derive *features* that may—or may not—prove trustworthy during bootstrapping.

Parses from a different domain. One might have a treebank for a *different* domain or genre of the target language.

One could simply include these trees in the initial supervised training, and hope that bootstrapping corrects any learned weights that are inappropriate to the target domain, as discussed above. In fact, McClosky et al. (2006) found a similar technique to be effective—though only in a model with a large feature space ("PCFG + reranking"), as we would predict.

However, another approach is to train a separate out-of-domain parser, and use this to generate *additional features* on the supervised and unsupervised in-domain data (Blitzer et al., 2006). Bootstrapping now teaches us where to trust the out-of-domain parser. If our basic model has 100 features, we could add features 101 through 200, where for example $f_{123}(e) = f_{23} \cdot \log \tilde{\Pr}(e)$ and $\tilde{\Pr}(e)$ is the posterior edge probability according to the out-of-domain parser. Learning that this feature has a high weight means learning to trust the out-of-domain parser's *decision* on edges where in-domain feature 23 fires. Even more sensibly, we could add features such as $f_{201}(e) = \sum_{i=1}^{10} \tilde{f}_i(e) \cdot \tilde{\theta}_i$, where $\tilde{\mathbf{f}}$ and $\tilde{\boldsymbol{\theta}}$ are the feature and weight vectors for the out-of-domain parser. Learning that this feature has a high weight means learning to trust the out-of-domain parser's *feature weights* for a particular *class* of features (those numbered 1 through 10). This addresses the intuition that some linguistic phenomena remain stable across domains.

Parses of translations. Suppose we have translations into English of *some* of our supervised or unsupervised sentences. Good probabilistic dependency parsers already exist for English, so we run one over the English translation. We can now derive many additional features on candidate edges on the target sentence. For example, dependency edges in the target language of the form $c \xrightarrow{poss} p$ (this denotes a child-to-parent dependency with label *possessor*) might often correspond to dependency paths in the English translation of the form $p' \xleftarrow{prep} \text{of} \xleftarrow{pobj} c'$. To discover whether this is so, we define a feature i by

$$f_i(c \xrightarrow{poss} p) \overset{\text{def}}{=} \log \sum_{c',p'} \begin{aligned}&(\Pr(c \text{ aligns with } c') \\ &\cdot \Pr(p \text{ aligns with } p') \\ &\cdot \Pr(p' \xleftarrow{prep} \text{of} \xleftarrow{pobj} c'))\end{aligned} \qquad (1)$$

where c', p' range over word tokens in the English translation, "of" is a literal English word, and the probabilities are posteriors provided by a probabilistic aligner and a probabilistic English parser. Note that this is a *single* feature (not a feature family parameterized by c, p). It scores any candidate edge on whether it is a $\xrightarrow{poss}$ edge that seems to align to an English $\xleftarrow{prep} \text{of} \xleftarrow{pobj}$ path.

This method is inspired by Hwa et al. (2005), who bootstrapped parsers for Spanish and Chinese by projecting dependencies from English translations and training a new parser on the resulting noisy treebank. They used only 1-best translations, 1-best alignments, dependency paths of length 1, and no labeled data in Spanish or Chinese.

Hwa et al. (2005) used a manually written postprocessor to correct some of the many incorrect projections. By contrast, our framework uses the projected dependencies only as one source of features. They may be overridden by other features in particular cases, and will be given a high weight only if they tend to agree with other features during bootstrapping. A similar *soft* projection of dependencies was used in supervised machine translation by Smith and Eisner (2006), who used a source sentence's dependency paths to *bias* the generation of its translation.

Note that these bilingual features will only fire on those supervised or unsupervised sentences for which we have an English translation. In particular, they will usually be unavailable on the test set. However, we hope that they will seed and facilitate the bootstrapping process, by helping us confidently parse some unsupervised sentences that we would not be able to confidently parse without an English translation.

Parses of comparable English sentences. World knowledge can be useful in parsing. Suppose you see a French sentence that contains *mangeons* and *pommes*, and you know that *manger=eat* and *pomme=apple*. You might reasonably guess that *pommes* is the direct object of *mangeons*, because you know that *apple* is a plausible direct object for *eat*. We can discover this last bit of world knowledge from comparable English text. Translation dictionaries can themselves be induced from comparable corpora (Schafer and Yarowsky, 2002; Schafer, 2006; Klementiev and Roth, 2006), or extracted from bitext or digitized versions of human-readable dictionaries if these are available.

The above inference pattern can be captured by features similar to those in equation (1). For example, one can define a feature j by

$$f_i(c \xrightarrow{poss} p) \overset{\text{def}}{=} \log \Pr(p' \xleftarrow{prep} \text{of} \xleftarrow{pobj} c' \\ \mid p' \text{ translates } p, c' \text{ translates } c) \qquad (2)$$

where each event in the event space is a pair (c', p') of same-sentence tokens in comparable English text, all pairs being equally likely. Thus, to estimate $\Pr(\cdot \mid \cdot)$, the denominator counts same-sentence token pairs (c', p') in the comparable English corpus that translate into the types (c, p), and the numerator counts such pairs that are *also* related by a $\xleftarrow{prep}$ of $\xleftarrow{pobj}$ path. Since the lexical translations and dependency paths are typically not labeled in the English corpus, a given pair must be counted fractionally according to its posterior probability of satisfying these conditions, given models of contextual translation and English parsing.[3]

2.4 Bootstrapping as Optimization

Section 2.2 assumed a relatively conventional kind of bootstrapping, where each iteration retrains the model on the examples where it is currently most confident. This kind of "confidence thresholding" has been popular in previous bootstrapping work (as cited in §2.2). It attempts to maintain high accuracy while gradually expanding coverage. The assumption is that throughout the training procedure, the parser's confidence is a trustworthy guide to its correctness. Different bootstrapping procedures use different learners, smoothing methods, confidence measures, and procedures for "forgetting" the labelings from previous iterations.

In his analysis of Yarowsky (1995), Abney (2004) formulates several variants of bootstrapping. These are shown to increase either the likelihood of the training data, or a lower bound on that likelihood. In particular, Abney defines a function K that is an upper bound on the *negative* log-likelihood, and shows his bootstrapping algorithms locally minimize K.

We now present a generalization of Abney's K function and relate it to another semi-supervised learning technique, entropy regularization (Brand, 1999; Grandvalet and Bengio, 2005; Jiao et al., 2006). Our experiments will tune the feature weight vector, $\boldsymbol{\theta}$, to minimize our function. We will do so simply by applying a *generic* function minimization method (stochastic gradient descent), rather than by crafting a new Yarowsky-style or Abney-style iterative procedure for our specific function.

Suppose we have examples x_i and corresponding possible labelings $y_{i,k}$. We are trying to learn a parametric model $p_{\boldsymbol{\theta}}(y_{i,k} \mid x_i)$. If $\tilde{p}(y_{i,k} \mid x_i)$ is a "labeling distribution" that reflects our uncertainty about the true labels, then our *expected* negative log-likelihood of the model is

$$\begin{aligned} K &\overset{\text{def}}{=} -\sum_i \sum_k \tilde{p}(y_{i,k} \mid x_i) \log p_{\boldsymbol{\theta}}(y_{i,k} \mid x_i) \\ &= \sum_i \sum_k \tilde{p}(y_{i,k}|x_i) \log \frac{\tilde{p}(y_{i,k}|x_i)}{p_{\boldsymbol{\theta}}(y_{i,k}|x_i)\tilde{p}(y_{i,k}|x_i)} \\ &= \sum_i D(\tilde{p}_i \| p_{\boldsymbol{\theta},i}) + H(\tilde{p}_i) \qquad (3) \end{aligned}$$

where $\tilde{p}_i(\cdot) \overset{\text{def}}{=} \tilde{p}(\cdot \mid x_i)$ and $p_{\boldsymbol{\theta},i}(\cdot) \overset{\text{def}}{=} p_{\boldsymbol{\theta}}(\cdot \mid x_i)$.

Note that K is a function not only of $\boldsymbol{\theta}$ but also

[3] Similarly, Jansche (2005) imputes "missing" trees by using comparable corpora.

of the labeling distribution $\tilde{p}$; a learner might be allowed to manipulate either in order to decrease K.

The summands of K in equation (3) can be divided into two cases, according to whether x_i is labeled or not. For the labeled examples $\{x_i : i \in L\}$, the labeling distribution $\tilde{p}_i$ is a point distribution that assigns all probability to the true, known label y_i^*. Then $H(\tilde{p}_i) = 0$. The total contribution of these examples to K simplifies to $\sum_{i \in L} - \log p_{\boldsymbol{\theta}}(y_i^* \mid x_i)$, i.e., just the negative log-likelihood on the labeled data.

But what is the labeling distribution for the unlabeled examples $\{x_i : i \notin L\}$? Abney simply uses a uniform distribution over labels (e.g., parses), to reflect that the label is unknown. If his bootstrapping algorithm "labels" x_i, then i moves into L and $H(\tilde{p}_i)$ is thereby reduced from maximal to 0. As a result, a method that labels the most confident examples may reduce K, and Abney shows that his method does so.

Our approach is different: we will take the labeling distribution $\tilde{p}_i$ to be our *actual* current belief $p_{\boldsymbol{\theta},i}$, and manipulate it through changing $\boldsymbol{\theta}$ rather than L. L remains the original set of *supervised* examples. The total contribution of the *unsupervised* examples to K then simplifies to $\sum_{i \notin L} H(p_{\boldsymbol{\theta},i})$.

We have no reason to believe that these two contributions (supervised and unsupervised) should be weighted equally. We thus introduce a multiplier γ to form the actual objective function that we minimize with respect to $\boldsymbol{\theta}$:[4]

$$-\sum_{i \in L} \log p_{\boldsymbol{\theta},i}(y_i^*) + \gamma \sum_{i \notin L}^{N} H(p_{\boldsymbol{\theta},i}) \qquad (4)$$

One may regard γ as a Lagrange multiplier that is used to constrain the classifier's uncertainty H to be low, as presented in the work on entropy regularization (Brand, 1999; Grandvalet and Bengio, 2005; Jiao et al., 2006).

Conventional bootstrapping retrains on the *most* confident unsupervised examples, making them more confident. Gradient descent on equation (4) essentially does the same, since unsupervised examples contribute to (4) only through H, and the shape of the H function means that it is most rapidly decreased by making the *most* confident unsupervised examples more confident.

Besides favoring models that are self-confident on the unlabeled data, the objective function (4) also explicitly asks the model to continue to get the correct answers on the initial supervised corpus. $1/\gamma$ controls the strength of this request. One could obtain a similar effect in conventional bootstrapping by upweighting the initial labeled corpus when retraining.

2.5 Online Learning

Minimizing equation (4) for parsing is more computationally intensive than in many other applications of bootstrapping, such as word sense disambiguation or document classification. With millions of features, our objective could take many iterations to converge to a local optimum, if we were only to update our parameter vector $\boldsymbol{\theta}$ after each iteration through a large unsupervised corpus.

For many machine learning problems over large datasets, online learning methods such as **stochastic gradient descent** (SGD) have been empirically observed to converge in fewer iterations (Bottou, 2003). In SGD, instead of taking an optimization step in the direction of the gradient calculated over all unsupervised training examples, we parse each example, calculate the gradient of the objective function evaluated on that example alone, and then take a small step downhill. The update rule is thus

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \cdot \nabla F^{(t)}(\boldsymbol{\theta}^{(t)}) \qquad (5)$$

where $\boldsymbol{\theta}^{(t)}$ is the parameter vector at time t, $F^{(t)}(\boldsymbol{\theta})$ is the objective function specialized to the time-t example, and $\eta > 0$ is a learning rate that we choose. We check for convergence after each pass through the example set.

2.6 Algorithms and Complexity

To evaluate equation (4), we need a conditional model of trees given a sentence x_i. We define one by exponentiating and normalizing the tree scores: $p_{\boldsymbol{\theta},i}(y_{i,k}) \stackrel{\text{def}}{=} \exp(\sum_{e \in y_{i,k}} \mathbf{f}(e) \cdot \boldsymbol{\theta})/Z_i$.

With exponentially many parses of x_i, does our objective function (4) now have prohibitive com-

[4] This function is not necessarily convex in $\boldsymbol{\theta}$, because of the addition of the entropy term (Jiao et al., 2006). One might try an annealing strategy: start γ at zero (where the function *is* convex) and gradually increase it, hoping to "ride" the global maximum. Although we could increase γ until the entropy term dominates the minimizations and we approach a completely deterministic classifier, it is preferable to use some labeled heldout data to evaluate a stopping criterion.

putational complexity? The complexity is actually similar to that of the inside algorithm for parsing. In fact, the first term of (4) for projective parsing is found by running the $O(n^3)$ inside algorithm on supervised data,[5] and its gradient is found by the corresponding $O(n^3)$ outside algorithm. For non-projective parsing, the analogy to the inside algorithm is the $O(n^3)$ "matrix-tree algorithm," which is dominated asymptotically by a matrix determinant (Smith and Smith, 2007; Koo et al., 2007; McDonald and Satta, 2007). The gradient of a determinant may be computed by matrix inversion, so evaluating the gradient again has the same $O(n^3)$ complexity as evaluating the function.

The second term of (4) is the Shannon entropy of the posterior distribution over parses. Computing this for projective parsing takes $O(n^3)$ time, using a dynamic programming algorithm that is closely related to the inside algorithm (Hwa, 2000).[6] For non-projective parsing, unfortunately, the runtime rises to $O(n^4)$, since it requires determinants of n distinct matrices (each incorporating a log factor in a different column; we omit the details). The gradient evaluation in both cases is again about as expensive as the function evaluation.

A convenient speedup is to replace Shannon entropy with Rényi entropy. The family of **Rényi entropy** measures is parameterized by α:

$$R_\alpha(p) \;=\; \frac{1}{1-\alpha}\log\left(\sum_y p(y)^\alpha\right) \qquad (6)$$

In our setting, where $p = p_{\boldsymbol{\theta},i}$, the events y are the possible parses $y_{i,k}$ of x_i. Observe that under our definition of p, $\sum_y p(y)^\alpha = \{\sum_y \exp[\sum_{e\in y}\mathbf{f}(e)\cdot(\alpha\boldsymbol{\theta})]\}/Z_i^\alpha$. We already have Z_i from running the inside algorithm, and we can find the numerator by running the inside algorithm again with $\boldsymbol{\theta}$ scaled by α. Thus with Rényi entropy, all computations and their gradients are $O(n^3)$—even in the non-projective case.

Rényi entropy is also a theoretically attractive generalization. It can be shown that $\lim_{\alpha\to 1} R_\alpha(p)$ is in fact the Shannon entropy $H(p)$ and that $\lim_{\alpha\to\infty} R_\alpha(p) = -\log\max_y p(y)$, i.e. the negative log probability of the modal or "Viterbi" label (Arndt, 2001; Karakos et al., 2007). The $\alpha = 2$ case, widely used as a measure of purity in decision tree learning, is often called the "Gini index." Finally, when $\alpha = 0$, we get the log of the number of labels, which equals the H(uniform distribution) that Abney used in equation (3).

3 Evaluation

For this paper, we performed some initial bootstrapping experiments on small corpora, using the features from (McDonald et al., 2005). After discussing experimental setup (§3.1), we look at the correlation of confidence with accuracy and with oracle likelihood, and at the fine-grained behaviour of models' dependency edge posteriors (§3.2). We then compare our confidence-maximizing bootstrapping to EM, which has been widely used in semi-supervised learning (§3.4). Section 3.3 presents overall bootstrapping accuracy.

3.1 Experimental Design

We bootstrapped non-projective parsers for languages assembled for the CoNLL dependency parsing competitions (Buchholz and Marsi, 2006). We selected German, Spanish, and Czech (Brants et al., 2002; Civit Torruella and Martí Antonín, 2002; Böhmová et al., 2003). After removing sentences more than 60 words long, we randomly divided each corpus into small seed sets of 100 and 1000 trees; development and test sets of 200 trees each; and an unlabeled training set from the rest.

These treebanks contain strict dependency trees, in the sense that their only nodes are the words and a distinguished root node. In the Czech dataset, more than one word can attach to the root; also, the trees in German, Spanish, and Czech may be non-projective. We use the `MSTParser` implementation described in McDonald et al. (2005) for feature extraction. Since our seed sets are so small, we extracted features from all edges in both the seed and the unlabeled parts of our training data, not just the edges annotated as correct. Since this produced many more features, we pruned our features to those with at least 10 occurrences over all edges.

[5] The numerator of $p_{\boldsymbol{\theta},i}(y_i^*)$ (see definition above) is trivial since y_i^* is a single known parse. But the denominator Z_i is a normalizing constant that sums over all parses; it is found by a dependency-parsing variant of the inside algorithm, following (Eisner, 1996).

[6] See also (Mann and McCallum, 2007) for similar results on conditional random fields.

		Correlation of			
		100-tree model		1000-tree model	
	Rényi α	Acc.	Xent.	Acc.	Xent.
(uniform, Abney)	0	-0.254	0.980	-0.180	0.937
	.5	-0.256	0.981	-0.203	0.955
(Shannon)	1	-0.260	0.983	-0.220	0.964
(Gini)	2	-0.266	0.985	-0.250	0.977
	5	-0.291	0.992	-0.304	0.990
	7	-0.301	0.993	-0.341	0.991
(Viterbi)	∞	-0.317	0.995	-0.326	0.992
	Xent.	-0.391	1.000	-0.410	1.000

Table 1: Correlation, on development sentences, of Rényi entropy with model accuracy and with cross-entropy ("Xent."). Since these are measures of uncertainty, we see a negative correlation. As α increases, we place more confidence in high-probability parses and correlate better with accuracy.

We used stochastic gradient descent first to minimize equation (4) on the labeled seed sets. Then we continued to optimize over the labeled and unlabeled data together. We tested for convergence using accuracy on development data.

3.2 Empirically Evaluating Entropy

Bootstrapping assumes that where the parser is confident, it tends to be correct. Standard bootstrapping methods retrain directly on confident links; similarly, our approach tries to make the parser even more confident on those links.

Is this assumption really true empirically? Yes: not only does confidence on unlabeled data correlate with cross-entropy, but both confidence and cross-entropy correlate well with accuracy. As we will see, some confidence measures correlate better than others. In particular, measures that are more peaked around the one-best prediction of the parser, as in Viterbi re-estimation, perform well.

If we train a non-projective German parser on small seed sets of 100 and 1000 trees, only, how well does its own confidence predict its performance? For 200 points—labeled development sentences—we measured the linear correlation of various Rényi entropies (6), normalized by sentence length, with tree accuracy (Table 1). We also measured how these normalized Rényi entropies correlate with the posterior log-probability the model assigns to the true parse (the cross-entropy).

Since Rényi entropy is a measure of uncertainty, we see a negative correlation with accuracy. This correlation strengthens as we raise α to ∞, so we might expect Viterbi re-estimation, or a differentiable objective function with a very high α, to perform best on held-out data. Note also that the cross-entropy, which looks at the true labels on the held-out data, does not itself correlate very much better with accuracy than the best *unsupervised* confidence measures. Finally, we see that Rényi entropies with higher α are more stable: when calculated for a model trained on more data, they improve their correlation with accuracy.

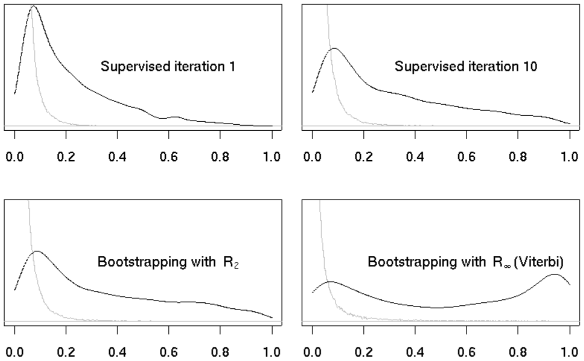


Figure 1: Posterior probability of correct and incorrect edges in German test data under various models. We show the distribution of posterior probabilities for correct edges, known from an oracle, in black and incorrect edges in gray. In the upper row, learning on an initial supervised set raises the posterior probability of correct edges while dragging along some incorrect edges. In the lower row, we see that adding unlabeled data with R_2 entropy continues the pattern of the supervised learner. R_∞ (Viterbi) training induces a second mode in correct posterior probabilities near 1 although it does shift more incorrect edges closer to 1.

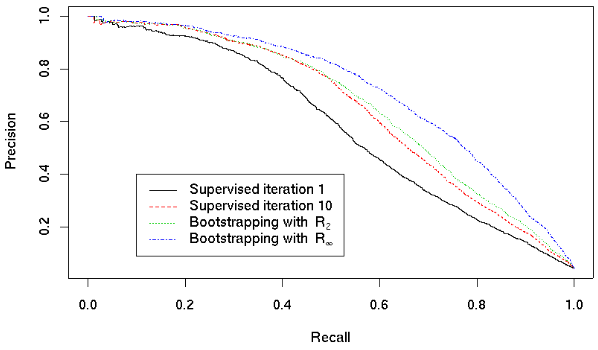


Figure 2: Precision-recall curves for selecting edges according to their posterior probabilities: better bootstrapping puts more area under the curve.

From tree confidence, we now turn to edge confidence: what is the posterior probability that a model assigns to each of the n^2 edges in the dependency graph? Figure 1 shows smoothed histograms of true edges (black) and false edges (gray) in held-out data, according to the posterior probabilities we assign to them. Since there are many more false edges, the figures are cropped to zoom in on the distribution of true edges. As we start training on the labeled seed set, the posterior probabilities of true edges move towards one; many false edges also get greater mass, but not to the same extent. As we add unlabeled data, we can see the different learning strategies of different confidence measures. R_2 gradually moves a few true and many fewer false edges towards 1, while R_∞ (Viterbi) learning is so confident as to induce a bimodal distribution in the posteriors of true edges. Figure 2 visualizes the same data as four precision-recall curves, which show how noisy the highest-confidence edges are, across a range of confidence thresholds. Although the very high precision end stays stable after 10 iterations on the seed set, the addition of unlabeled data puts more area under the curve. Again, R_∞ dominates R_2.

3.3 Bootstrapping Results

We performed bootstrapping experiments on the full CoNLL sets for Czech, German, and Spanish using the non-projective model from McDonald et al. (2005). Performance confirms the results of our analysis above (Table 2). Adding unlabeled data improves performance over that of the seed set, with the exception of the Czech data with R_2 bootstrapping. As we saw in §3.2, bootstrapping with R_∞ dominates bootstrapping with R_2 confidence. For comparison, we also show the results obtained by supervised training on the combined seed and unlabeled sets. Recall that we did not use the tree annotations to perform feature selection; models trained with only supported features ought to perform better.

Although we see statistically significant improvements (at the .05 level on a paired permutation test), the quality of the parsers is still quite poor, in contrast to other applications of bootstrapping which "rival supervised methods" (Yarowsky, 1995). Almost certainly, the CoNLL datasets, comprising at most some tens of thousands of sentences per language, are too small to afford qualitative improvements. Also, at these relatively small training sizes, our preliminary attempts to leverage comparable English corpora did not improve performance.

		% accuracy		
	Seed trees	$\alpha = 0$	2	∞
Czech	100	56.1	54.8	**58.3**
	1000	68.1	68.2	68.2
	71468	77.9	–	–
German	100	60.9	**62.4**	**65.3**
	1000	74.6	74.5	**75.0**
	37745	86.0	–	–
Spanish	100	63.6	**64.1**	**64.4**
	2786	76.6	–	–

Table 2: Dependency accuracy of the McDonald model on 200 test sentences. When $\alpha = 0$, training only occurs on the supervised seed data. As α increases, we train based on confidence in our model's analysis of the unlabeled data. **Boldface** results are the best in their rows in a permutation test at the .05 level.

What features were learned, and how dependent is performance on the seed set? We analyzed the performance of German bootstrapping on a development set (Table 3). Using the parameters at the last iteration of supervised training on the seed set as a baseline, we tried updating to their bootstrapped values the weights of only those features that occurred in the seed set. This achieved nearly the same accuracy as updating all the features. As one would expect, using only the non-seed features' weights performs abysmally. This might be the case simply because the seed set is likely to contain frequently occurring features. If, however, we use only the features occurring in an alternate training set of the same size (100 sentences), we get much worse performance. These results indicate that our bootstrapped parser is still heavily dependent on the features that happened to fire in the seed set; we have not "forgotten" our initial conditions. Similar experiments show that unlexicalized features contribute the most to bootstrapping performance. Since in our log-linear models features have been trained to work together, we must not put too much weight on these ablation results. These experiments do, however, suggest that bootstrapping improved our results by refining the values of known, non-lexicalized features.

3.4 Comparison with EM

Perhaps the most popular statistical method for learning from incomplete data is the EM algorithm (Dempster et al., 1977). Since we cannot try EM on McDonald's conditional model, we ran some pilot experiments using the generative dependency model with valence (DMV) of Klein and Manning (2004). As in their experiments, and unlike the other experiments in the current paper, we restricted ourselves

Updated	M feat.	acc.	Updated	M feat.	acc.
all	15.5	64.3	none	0	60.9
seed	1.4	64.1	non-seed	14.1	44.7
non-lexical	3.5	64.4	lexical	12.0	59.9
non-bilex.	12.6	64.4	bilexical	2.9	61.0

Table 3: Using all features, dependency accuracy on German development data rose to 64.3% on bootstrapping. We show the contribution of different feature splits to the performance of this final model. For example, although this model was trained by updating all 15.5M feature weights, it performs as well if we *then* keep only the 1.4M features that appeared at least once in the seed set, zeroing out the weights of the others. We do as well as the full feature set if we keep only the 3.5M non-lexicalized features.

		% accuracy		
	train	Bulg.	German	Spanish
supervised	ML	74.2	80.0	71.3
	CL	77.5	79.3	75.0
semi-supervised	EM	58.6	58.8	68.4
	Conf.	80.0	80.5	76.7

Table 4: Dependency accuracy of the DMV model (Klein and Manning, 2004). Maximizing confidence using R_1 (Shannon) entropy improved performance over its own conditional likelihood (CL) baseline and over maximum likelihood (ML). EM degraded its ML baseline. Since these models were only trained and tested on sentences of 10 words or fewer, accuracy is much higher than the full results in Table 2.

to sentences of ten words or fewer and to part-of-speech sequences alone, without any lexical information. Since the DMV models projective trees, we ran experiments on three CoNLL corpora that had augmented their primary non-projective parses with alternate projective annotations: Bulgarian (Simov et al., 2005), German, and Spanish.

We performed supervised maximum likelihood and conditional likelihood estimation on a seed set of 100 sentences for each language. These models respectively initialized EM and confidence training on unlabeled data. We see (Table 4) that EM degrades the performance of its ML baseline. Merialdo (1994) saw a similar degradation over small (and large) seed sets in HMM POS tagging. We tried fixing and not fixing the feature expectations on the seed set during EM and show the former, better numbers. Confidence maximization improved over both its own conditional likelihood initializer and also over ML. We selected optimal smoothing parameters for all models and optimal α (equation (6)) and γ (equation (4)) for the confidence model on labeled held-out data.

4 Future Work

We hypothesize that qualitatively better bootstrapping results will require much larger unlabeled data sets. In scaling up bootstrapping to larger unlabeled training sets, we must carefully weight tradeoffs between expanding coverage and introducing noise from out-of-domain data. We could also better exploit the data we have with richer models of syntax. In supervised dependency parsing, second-order edge features provide improvements (McDonald and Pereira, 2006; Riedel and Clarke, 2006); moreover, the feature-based approach is not limited to dependency parsing. Similar techniques could score parses in other formalisms, such as CFG or TAG. In this case, $\mathbf{f}$ extracts features from each of the derivation tree's rewrite rules (CFG) or elementary trees (TAG). In lexicalized formalisms, $\mathbf{f}$ will still be able to score lexical dependencies that are implicitly represented in the parse. Finally, we want to investigate whether larger training sets will provide traction for sparser cross-lingual and cross-domain features.

5 Conclusions

Feature-rich dependency models promise to help bootstrapping by providing many redundant features for the learner, and they can also cleanly incorporate cross-domain and cross-language information.

We explored bootstrapping feature-rich non-projective dependency parsers for Czech, German, and Spanish. Our bootstrapping method maximizes a linear combination of likelihood and confidence. In initial experiments on small datasets, this surpassed EM for training a simple feature-poor generative model, and also improved the performance of a feature-rich, conditionally estimated model where EM could not easily have been applied. For our models and training sets, more peaked measures of confidence, measured by Rényi entropy, outperformed smoother ones.

Acknowledgments

The authors thank the anonymous reviewers, Noah A. Smith, and Keith Hall for helpful comments, and Ryan McDonald for making his parsing code publicly available. This work was supported in part by NSF ITR grant IIS-0313193.

References

Steven Abney. 2004. Understanding the Yarowsky algorithm. *CL*, 30(3):365–395.

Cristoph Arndt. 2001. *Information Measures*. Springer.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128.

A. Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*, chapter 7. Kluwer.

Léon Bottou. 2003. Stochastic learning. In *Advanced Lectures in Machine Learning*, pages 146–168. Springer.

Matthew E. Brand. 1999. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *TLT*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Civit Torruella and M. A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *TLT*.

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *EMNLP*, pages 232–241.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING*, pages 340–345.

H. N. Gabow, Z. Galil, T. H. Spencer, and R. E. Tarjan. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *ACL*, pages 961–968.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *NIPS*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:311–325.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *EMNLP*, pages 45–52.

Martin Jansche. 2005. Treebank transfer. In *IWPT*.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *COLING/ACL*, pages 209–216.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Carey E. Priebe. 2007. Cross-instance tuning of unsupervised document clustering algorithms. In *HLT-NAACL*, pages 252–259.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*, pages 479–486.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *COLING-ACL*, pages 817–824.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured prediction models via the Matrix-Tree Theorem. In *EMNLP-CoNLL*.

Gideon S. Mann and Andrew McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *HLT-NAACL*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *EMNLP*, pages 44–52, July.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *ACL*, pages 337–344.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *IWPT*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL*, pages 91–98.

Bernardo Merialdo. 1994. Tagging English text with a probabilistic model. *CL*, 20(2):155–72.

Fuchun Peng, Ralph Weischedel, Ana Licuanan, and Jinxi Xu. 2005. Combining deep linguistics analysis and surface pattern learning: A hybrid approach to Chinese definitional question answering. In *HLT-EMNLP*, pages 307–314.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP*, pages 129–137.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*.

Charles Schafer. 2006. *Translation Discovery Using Diverse Smilarity Measures*. Ph.D. thesis, Johns Hopkins University.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*. Kluwer.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *EMNLP-CoNLL*.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *EACL*.

Ralph Weischedel. 2004. Extracting dynamic evidence networks. Technical Report AFRL-IF-RS-TR-2004-246, BBN Technologies, Cambridge, MA, December.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *ACL*, pages 207–216.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pages 189–196.

Online Learning of Relaxed CCG Grammars for Parsing to Logical Form

Luke S. Zettlemoyer and **Michael Collins**
MIT CSAIL
lsz@csail.mit.edu,mcollins@csail.mit.edu

Abstract

We consider the problem of learning to parse sentences to lambda-calculus representations of their underlying semantics and present an algorithm that learns a weighted combinatory categorial grammar (CCG). A key idea is to introduce non-standard CCG combinators that relax certain parts of the grammar—for example allowing flexible word order, or insertion of lexical items—with learned costs. We also present a new, online algorithm for inducing a weighted CCG. Results for the approach on ATIS data show 86% F-measure in recovering fully correct semantic analyses and 95.9% F-measure by a partial-match criterion, a more than 5% improvement over the 90.3% partial-match figure reported by He and Young (2006).

1 Introduction

Recent work (Mooney, 2007; He and Young, 2006; Zettlemoyer and Collins, 2005) has developed learning algorithms for the problem of mapping sentences to underlying semantic representations. In one such approach (Zettlemoyer and Collins, 2005) (ZC05), the input to the learning algorithm is a training set consisting of sentences paired with lambda-calculus expressions. For instance, the training data might contain the following example:

Sentence: list flights to boston
Logical Form: $\lambda x.flight(x) \wedge to(x, boston)$

In this case the lambda-calculus expression denotes the set of all flights that land in Boston. In ZC05 it is assumed that training examples do not include additional information, for example parse trees or other derivations. The output from the learning algorithm is a combinatory categorial grammar (CCG), together with parameters that define a log-linear distribution over parses under the grammar. Experiments show that the approach gives high accuracy on two database-query problems, introduced by Zelle and Mooney (1996) and Tang and Mooney (2000).

a) on may four atlanta to denver delta flight 257
$\lambda x.month(x, may) \wedge day\_number(x, fourth) \wedge$
$from(x, atlanta) \wedge to(x, denver) \wedge$
$airline(x, delta\_air\_lines) \wedge flight(x) \wedge$
$flight\_number(x, 257)$

b) show me information on american airlines from fort worth texas to philadelphia
$\lambda x.airline(x, american\_airlines) \wedge$
$from(x, fort\_worth) \wedge to(x, philadelphia)$

c) okay that one's great too now we're going to go on april twenty second dallas to washington the latest nighttime departure one way
$argmax(\lambda x.flight(x) \wedge from(x, dallas) \wedge$
$to(x, washington) \wedge month(x, april) \wedge$
$day\_number(x, 22) \wedge during(x, night) \wedge$
$one\_way(x), \lambda y.depart\_time(y))$

Figure 1: Three sentences from the ATIS domain.

The use of a detailed grammatical formalism such as CCG has the advantage that it allows a system to handle quite complex semantic effects, such as coordination or scoping phenomena. In particular, it allows us to leverage the considerable body of work on semantics within these formalisms, for example see Carpenter (1997). However, a grammar based on a formalism such as CCG can be somewhat rigid, and this can cause problems when a system is faced with spontaneous, unedited natural language input, as is commonly seen in natural language interface applications. For example, consider the sentences shown in figure 1, which were taken from the ATIS travel-planning domain (Dahl et al., 1994). These sentences exhibit characteristics which present significant challenges to the approach of ZC05. For ex-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 678–687, Prague, June 2007. ©2007 Association for Computational Linguistics

ample, the sentences have quite flexible word order, and include telegraphic language where some words are effectively omitted.

In this paper we describe a learning algorithm that retains the advantages of using a detailed grammar, but is highly effective in dealing with phenomena seen in spontaneous natural language, as exemplified by the ATIS domain. A key idea is to extend the approach of ZC05 by allowing additional nonstandard CCG combinators. These combinators relax certain parts of the grammar—for example allowing flexible word order, or insertion of lexical items—with learned costs for the new operations. This approach has the advantage that it can be seamlessly integrated into CCG learning algorithms such as the algorithm described in ZC05.

A second contribution of the work is a new, online algorithm for CCG learning. The approach involves perceptron training of a model with hidden variables. In this sense it is related to the algorithm of Liang et al. (2006). However it has the additional twist of also performing grammar induction (lexical learning) in an online manner. In our experiments, we show that the new algorithm is considerably more efficient than the ZC05 algorithm; this is important when training on large training sets, for example the ATIS data used in this paper.

Results for the approach on ATIS data show 86% F-measure accuracy in recovering fully correct semantic analyses, and 95.9% F-measure by a partial-match criterion described by He and Young (2006). The latter figure contrasts with a figure of 90.3% for the approach reported by He and Young (2006).[1] Results on the Geo880 domain also show an improvement in accuracy, with 88.9% F-measure for the new approach, compared to 87.0% F-measure for the method in ZC05.

2 Background

2.1 Semantics

Training examples in our approach consist of sentences paired with lambda-calculus expressions. We use a version of the lambda calculus that is closely related to the one presented by Carpenter (1997). There are three basic types: t, the type of truth values; e, the type for entities; and r, the type for real numbers. Functional types are defined by specifying their input and output types, for example $\langle e, t\rangle$ is the type of a function from entities to truth values. In general, declarative sentences have a logical form of type t. Question sentences generally have functional types.[2] Each expression is constructed from constants, logical connectors, quantifiers and lambda functions.

2.2 Combinatory Categorial Grammars

Combinatory categorial grammar (CCG) is a syntactic theory that models a wide range of linguistic phenomena (Steedman, 1996; Steedman, 2000). The core of a CCG grammar is a lexicon Λ. For example, consider the lexicon

flights	:=	$N : \lambda x.flight(x)$
to	:=	$(N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x, y)$
boston	:=	$NP : boston$

Each entry in the lexicon is a pair consisting of a word and an associated category. The category contains both syntactic and semantic information. For example, the first entry states that the word *flights* can have the category $N : \lambda x.flight(x)$. This category consists of a syntactic type N, together with the semantics $\lambda x.flight(x)$. In general, the semantic entries for words in the lexicon can consist of any lambda-calculus expression. Syntactic types can either be simple types such as N, NP, or S, or can be more complex types that make use of slash notation, for example $(N\backslash N)/NP$.

CCG makes use of a set of *combinators* which are used to combine categories to form larger pieces of syntactic and semantic structure. The simplest such rules are the *functional application* rules:

$$\begin{array}{lll l} A/B : f \quad B : g & \Rightarrow & A : f(g) & (>) \\ B : g \quad A\backslash B : f & \Rightarrow & A : f(g) & (<) \end{array}$$

The first rule states that a category with syntactic type A/B can be combined with a category to the right of syntactic type B to create a new category of type A. It also states that the new semantics will be formed by applying the function f to the expression g. The second rule handles arguments to the left. Using these rules, we can parse the

[1] He and Young (2006) do not give results for recovering fully correct parses.

[2] For example, many question sentences have semantics of type $\langle e, t\rangle$, as in $\lambda x.flight(x) \wedge to(x, boston)$.

following phrase to create a new category of type N:

$$
\begin{array}{ccc}
\text{flights} & \text{to} & \text{boston} \\
\hline
N & (N\backslash N)/NP & NP \\
\lambda x.flight(x) & \lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y) & \textit{boston} \\
\cline{2-3}
 & \multicolumn{2}{c}{(N\backslash N) \quad >} \\
 & \multicolumn{2}{c}{\lambda f.\lambda x.f(x) \wedge to(x, boston)} \\
\hline
\multicolumn{3}{c}{N \quad <} \\
\multicolumn{3}{c}{\lambda x.flight(x) \wedge to(x, boston)}
\end{array}
$$

The top-most parse operations pair each word with a corresponding category from the lexicon. The later steps are labeled $->$ (for each instance of forward application) or $-<$ (for backward application).

A second set of combinators in CCG grammars are the rules of *functional composition*:

$$
\begin{array}{llll}
A/B : f \quad B/C : g & \Rightarrow & A/C : \lambda x.f(g(x)) & (> \mathbf{B}) \\
B\backslash C : g \quad A\backslash B : f & \Rightarrow & A\backslash C : \lambda x.f(g(x)) & (< \mathbf{B})
\end{array}
$$

These rules allow for an unrestricted notion of constituency that is useful for modeling coordination and other linguistic phenomena. As we will see, they also turn out to be useful when modeling constructions with relaxed word order, as seen frequently in domains such as ATIS.

In addition to the application and composition rules, we will also make use of type raising and coordination combinators. A full description of these combinators goes beyond the scope of this paper. Steedman (1996; 2000) presents a detailed description of CCG.

2.3 Log-Linear CCGs

We can generalize CCGs to weighted, or probabilistic, models as follows. Our models are similar to several other approaches (Ratnaparkhi et al., 1994; Johnson et al., 1999; Lafferty et al., 2001; Collins, 2004; Taskar et al., 2004). We will write x to denote a sentence, and y to denote a CCG parse for a sentence. We use GEN$(x; \Lambda)$ to refer to all possible CCG parses for x under some CCG lexicon Λ. We will define $\mathbf{f}(x, y) \in \mathbb{R}^d$ to be a d-dimensional *feature–vector* that represents a parse tree y paired with an input sentence x. In principle, $\mathbf{f}$ could include features that are sensitive to arbitrary substructures within the pair (x, y). We will define $\mathbf{w} \in \mathbb{R}^d$ to be a parameter vector. The optimal parse for a sentence x under parameters $\mathbf{w}$ and lexicon Λ is then defined as

$$y^*(x) = \arg \max_{y \in \text{GEN}(x;\Lambda)} \mathbf{w} \cdot \mathbf{f}(x, y) \quad .$$

Assuming sufficiently local features[3] in $\mathbf{f}$, search for y^* can be achieved using dynamic-programming-style algorithms, typically with some form of beam search.[4] Training a model of this form involves learning the parameters $\mathbf{w}$ and potentially also the lexicon Λ. This paper focuses on a method for learning a $(\mathbf{w}, \Lambda)$ pair from a training set of sentences paired with lambda-calculus expressions.

2.4 Zettlemoyer and Collins 2005

We now give a description of the approach of Zettlemoyer and Collins (2005). This method will form the basis for our approach, and will be one of the baseline models for the experimental comparisons.

The input to the ZC05 algorithm is a set of training examples (x_i, z_i) for $i = 1 \ldots n$. Each x_i is a sentence, and each z_i is a corresponding lambda-expression. The output from the algorithm is a pair $(\mathbf{w}, \Lambda)$ specifying a set of parameter values, and a CCG lexicon. Note that for a given training example (x_i, z_i), there may be many possible parses y which lead to the correct semantics z_i.[5] For this reason the training problem is a *hidden-variable* problem, where the training examples contain only partial information, and the CCG lexicon and parse derivations must be learned without direct supervision.

A central part of the ZC05 approach is a function GENLEX(x, z) which maps a sentence x together with semantics z to a set of potential lexical entries. The function GENLEX is defined through a set of rules—see figure 2—that consider the expression z, and generate a set of categories that may help in building the target semantics z. An exhaustive set of lexical entries is then generated by taking all categories generated by the GENLEX rules, and pairing them with all possible sub-strings of the sentence x. Note that our lexicon can contain multi-word entries, where a multi-word string such as *New York* can be paired with a CCG category. The final out-

[3] For example, features which count the number of lexical entries of a particular type, or features that count the number of applications of a particular CCG combinator.

[4] In our experiments we use a parsing algorithm that is similar to a CKY-style parser with dynamic programming. Dynamic programming is used but each entry in the chart maintains a full semantic expression, preventing a polynomial-time algorithm; beam search is used to make the approach tractable.

[5] This problem is compounded by the fact that the lexicon is unknown, so that many of the possible hidden derivations involve completely spurious lexical entries.

Rules		Example categories produced from the logical form $\arg\max(\lambda x.flight(x) \land from(x, boston), \lambda x.cost(x))$
Input Trigger	Output Category	
constant c	$NP : c$	$NP : boston$
arity one predicate p	$N : \lambda x.p(x)$	$N : \lambda x.flight(x)$
arity one predicate p	$S\backslash NP : \lambda x.p(x)$	$S\backslash NP : \lambda x.flight(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y, x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.from(y, x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x, y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.from(x, y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \land g(x)$	$N/N : \lambda g.\lambda x.flight(x) \land g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x, c) \land g(x)$	$N/N : \lambda g.\lambda x.from(x, boston) \land g(x)$
arity two predicate p_2	$(N\backslash N)/NP : \lambda y.\lambda g.\lambda x.p_2(x, y) \land g(x)$	$(N\backslash N)/NP : \lambda y.\lambda g.\lambda x.from(x, y) \land g(x)$
an $\arg\max$ / $\min$ with second argument arity one function f	$NP/N : \lambda g.\arg\max / \min(g, \lambda x.f(x))$	$NP/N : \lambda g.\arg\max(g, \lambda x.cost(x))$
arity one function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.cost(x)$
arity one function f	$(N\backslash N)/NP : \lambda y.\lambda f.\lambda x.g(x) \land f(x) >/< y$	$(N\backslash N)/NP : \lambda y.\lambda f.\lambda x.g(x) \land cost(x) > y$
no trigger	$S/NP : \lambda x.x, \quad S/N : \lambda f.\lambda x.f(x)$	$S/NP : \lambda x.x, \quad S/N : \lambda f.\lambda x.f(x)$

Figure 2: Rules used in GENLEX. Each row represents a rule. The first column lists the triggers that identify some sub-structure within a logical form. The second column lists the category that is created. The third column lists categories that are created when the rule is applied to the logical form at the top of this column. We use the 10 rules described in ZC05 and add two new rules, listed in the last two rows above. This first new rule is instantiated for greater than (>) and less than (<) comparisions. The second new rule has no trigger; it is always applied. It generates categories that are used to learn lexical entries for semantically vacuous sentence prefixes such as the phrase *show me information on* in the example in figure 1(b).

put from GENLEX(x, z) is a large set of potential lexical entries, with the vast majority of those entries being spurious. The algorithm in ZC05 embeds GENLEX within an overall learning approach that simultaneously selects a small subset of all entries generated by GENLEX and estimates parameter values **w**. Zettlemoyer and Collins (2005) present more complete details. In section 4.2 we describe a new, online algorithm that uses GENLEX.

3 Parsing Extensions: Combinators

This section describes a set of CCG combinators which we add to the conventional CCG combinators described in section 2.2. These additional combinators are natural extensions of the forward application, forward composition, and type-raising rules seen in CCG. We first describe a set of combinators that allow the parser to significantly relax constraints on word order. We then describe a set of type-raising rules which allow the parser to cope with telegraphic input (in particular, missing function words). In both cases these additional rules lead to significantly more parses for any sentence x given a lexicon Λ. Many of these parses will be suspect from a linguistic perspective; broadening the set of CCG combinators in this way might be considered a dangerous move. However, the learning algorithm in our approach can learn weights for the new rules, effectively allowing the model to learn to use them only in appropriate contexts; in the experiments we show that the rules are highly effective additions when used within a weighted CCG.

3.1 Application and Composition Rules

The first new combinators we consider are the *relaxed functional application* rules:

$$\begin{array}{llcll} A\backslash B : f & B : g & \Rightarrow & A : f(g) & (\gtrsim) \\ B : g & A/B : f & \Rightarrow & A : f(g) & (\lesssim) \end{array}$$

These are variants of the original application rules, where the slash direction on the principal categories (A/B or $A\backslash B$) is reversed.[6] These rules allow simple reversing of regular word order, for example

$$\begin{array}{cc} \text{flights} & \text{one way} \\ \hline N & N/N \\ \lambda x.flight(x) & \lambda f.\lambda x.f(x) \land one\_way(x) \\ \hline \multicolumn{2}{c}{N} \\ \multicolumn{2}{c}{\lambda x.flight(x) \land one\_way(x)} \end{array} \lesssim$$

Note that we can recover the correct analysis for this fragment, with the same lexical entries as those used for the conventional word order, *one-way flights*.

A second set of new combinators are the *relaxed functional composition* rules:

$$\begin{array}{llcll} A\backslash B : f & B/C : g & \Rightarrow & A/C : \lambda x.f(g(x)) & (\gtrsim \mathbf{B}) \\ B\backslash C : g & A/B : f & \Rightarrow & A\backslash C : \lambda x.f(g(x)) & (\lesssim \mathbf{B}) \end{array}$$

These rules are variantions of the standard functional composition rules, where the slashes of the principal categories are reversed.

[6] Rules of this type are non-standard in the sense that they violate Steedman's Principle of Consistency (2000); this principle states that rules must be consistent with the slash direction of the principal category. Steedman (2000) only considers rules that do not violate this principle—for example, crossed composition rules, which we consider later, and which Steedman also considers, do not violate this principle.

An important point is that that these new composition and application rules can deal with quite flexible word orders. For example, take the fragment *to washington the latest flight*. In this case the parse is

$$\begin{array}{cccl}
\text{to washington} & \text{the latest} & \text{flight} & \\
\hline
N\backslash N & NP/N & N & \\
\lambda f.\lambda x.f(x)\wedge & \lambda f.\arg\max(f, & \lambda x.flight(x) & \\
to(x, washington) & \lambda y.depart\_time(y)) & & \\
\cline{1-2}
\multicolumn{2}{c}{} & & \lesssim_{\mathbf{B}} \\
\multicolumn{2}{c}{NP\backslash N} & & \\
\multicolumn{2}{c}{\lambda f.\arg\max(\lambda x.f(x)\wedge} & & \\
\multicolumn{2}{c}{to(x, washington), \lambda y.depart\_time(y))} & & \\
\hline
\multicolumn{3}{c}{} & \gtrsim \\
\multicolumn{3}{c}{NP} & \\
\multicolumn{3}{c}{\arg\max(\lambda x.flight(x)\wedge to(x, washington),} & \\
\multicolumn{3}{c}{\lambda y.depart\_time(y))} &
\end{array}$$

Note that in this case the substring *the latest* has category NP/N, and this prevents a naive parse where *the latest* first combines with *flight*, and *to washington* then combines with *the latest flight*. The functional composition rules effectively allow *the latest* to take scope over *flight* and *to washington*, in spite of the fact that *the latest* appears between the two other sub-strings. Examples like this are quite frequent in domains such as ATIS.

We add features in the model which track the occurrences of each of these four new combinators. Specifically, we have four new features in the definition of **f**; each feature tracks the number of times one of the combinators is used in a CCG parse. The model learns parameter values for each of these features, allowing it to learn to penalise these rules to the correct extent.

3.2 Additional Rules of Type-Raising

We now describe new CCG operations designed to deal with cases where words are in some sense missing in the input. For example, in the string *flights Boston to New York*, one style of analysis would assume that the preposition *from* had been deleted from the position before *Boston*.

The first set of rules is generated from the following *role-hypothesising type shifting* rules template:

$$NP : c \quad \Rightarrow \quad N\backslash N : \lambda f.\lambda x.f(x)\wedge p(x, c) \qquad (\mathbf{T_R})$$

This rule can be applied to any NP with semantics c, and any arity-two function p such that the second argument of p has the same type as c. By "any" arity-two function, we mean any of the arity-two functions seen in training data. We define features within the feature-vector **f** that are sensitive to the number of times these rules are applied in a parse; a separate feature is defined for each value of p.

In practice, in our experiments most rules of this form have p as the semantics of some preposition, for example *from* or *to*. A typical example of a use of this rule would be the following:

$$\begin{array}{cccl}
\text{flights} & \text{boston} & \text{to new york} & \\
\hline
N & NP & N\backslash N & \\
\lambda x.flight(x) & bos & \lambda f.\lambda x.f(x) & \\
 & & \wedge to(x, new\_york) & \\
\cline{2-2}
 & & & \mathbf{T_R} \\
 & N\backslash N & & \\
 & \lambda f.\lambda x.f(x)\wedge from(x, bos) & & \\
\cline{1-2}
\multicolumn{2}{c}{} & & < \\
\multicolumn{2}{c}{N} & & \\
\multicolumn{2}{c}{\lambda f.\lambda x.flight(x)\wedge from(x, bos)} & & \\
\hline
\multicolumn{3}{c}{} & < \\
\multicolumn{3}{c}{N} & \\
\multicolumn{3}{c}{\lambda x.flight(x)\wedge to(x, new\_york)\wedge from(x, bos)} &
\end{array}$$

The second rule we consider is the *null-head type shifting* rule:

$$N\backslash N : f \quad \Rightarrow \quad N : f(\lambda x.true) \qquad (\mathbf{T_N})$$

This rule allows parses of fragments such as *American Airlines from New York*, where there is again a word that is in some sense missing (it is straightforward to derive a parse for *American Airlines flights from New York*). The analysis would be as follows:

$$\begin{array}{ccl}
\text{American Airlines} & \text{from New York} & \\
\hline
N/N & N\backslash N & \\
\lambda f.\lambda x.f(x)\wedge airline(x, aa) & \lambda f.\lambda x.f(x)\wedge from(x, new\_york) & \\
\cline{2-2}
 & & \mathbf{T_N} \\
 & N & \\
 & \lambda x.from(x, new\_york) & \\
\hline
\multicolumn{2}{c}{} & > \\
\multicolumn{2}{c}{N} & \\
\multicolumn{2}{c}{\lambda x.airline(x, aa)\wedge from(x, new\_york)} &
\end{array}$$

The new rule effectively allows the prepositional phrase *from New York* to type-shift to an entry with syntactic type N and semantics $\lambda x.from(x, new\_york)$, representing the set of all things from New York.[7]

We introduce a single additional feature which counts the number of times this rule is used.

3.3 Crossed Composition Rules

Finally, we include *crossed functional composition* rules:

$$\begin{array}{llcll}
A/B : f & B\backslash C : g & \Rightarrow & A\backslash C : \lambda x.f(g(x)) & (>\mathbf{B}_\times) \\
B/C : g & A\backslash B : f & \Rightarrow & A/C : \lambda x.f(g(x)) & (<\mathbf{B}_\times)
\end{array}$$

These rules are standard CCG operators but they were not used by the parser described in ZC05. When used in unrestricted contexts, they can significantly relax word order. Again, we address this

[7] Note that we do not analyze this prepositional phrase as having the semantics $\lambda x.flight(x)\wedge from(x, new\_york)$—although in principle this is possible—as the $flight(x)$ predicate is not necessarily implied by this utterance.

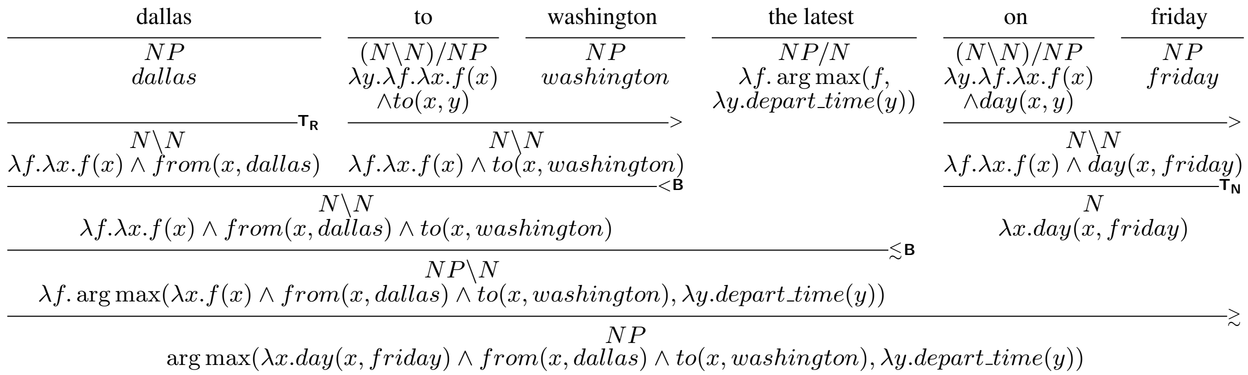

dallas
to
washington
the latest
on
friday
NP dallas
(N\N)/NP λy.λf.λx.f(x) ∧to(x,y)
NP washington
NP/N λf.arg max(f, λy.depart_time(y))
(N\N)/NP λy.λf.λx.f(x) ∧day(x,y)
NP friday
T_R
>
>
N\N λf.λx.f(x) ∧ from(x,dallas)
N\N λf.λx.f(x) ∧ to(x,washington)
N\N λf.λx.f(x) ∧ day(x,friday)
<B
T_N
N\N λf.λx.f(x) ∧ from(x,dallas) ∧ to(x,washington)
N λx.day(x,friday)
≲B
NP\N λf.arg max(λx.f(x) ∧ from(x,dallas) ∧ to(x,washington), λy.depart_time(y))
≳
NP arg max(λx.day(x,friday) ∧ from(x,dallas) ∧ to(x,washington), λy.depart_time(y))

Figure 3: A parse with the flexible parser.

problem by introducing features that count the number of times they are used in a parse.[8]

3.4 An Example

As a final point, to see how these rules can interact in practice, see figure 3. This example demonstrates the use of the relaxed application and composition rules, as well as the new type-raising rules.

4 Learning

This section describes an approach to learning in our model. We first define the features used and then describe a new online learning algorithm for the task.

4.1 Features in the Model

Section 2.3 described the use of a function $\mathbf{f}(x, y)$ which maps a sentence x together with a CCG parse y to a feature vector. As described in section 3, we introduce features for the new CCG combinators. In addition, we follow ZC05 in defining features which track the number of times each lexical item in Λ is used. For example, we would have one feature tracking the number of times the lexical entry $flights := N : \lambda x.flights(x)$ is used in a parse, and similar features for all other members of Λ.

Finally, we introduce new features which directly consider the semantics of a parse. For each predicate f seen in training data, we introduce a feature that counts the number of times f is conjoined with itself at some level in the logical form. For example, the expression $\lambda x.flight(x) \wedge from(x, new\_york) \wedge from(x, boston)$ would trigger the new feature for the $from$ predicate signaling that the logical-form describes flights with more than one origin city. We introduce similar features which track disjunction as opposed to conjunction.

4.2 An Online Learning Algorithm

Figure 4 shows a learning algorithm that takes a training set of (x_i, z_i) pairs as input, and returns a weighted CCG (i.e., a pair $(\mathbf{w}, \Lambda)$) as its output. The algorithm is *online*, in that it visits each example in turn, and updates both $\mathbf{w}$ and Λ if necessary. In Step 1 on each example, the input x_i is parsed. If it is parsed correctly, the algorithm immediately moves to the next example. In Step 2, the algorithm temporarily introduces all lexical entries seen in GENLEX(x_i, z_i), and finds the highest scoring parse that leads to the correct semantics z_i. A small subset of GENLEX(x_i, z_i)—namely, only those lexical entries that are contained in the highest scoring parse—are added to Λ. In Step 3, a simple perceptron update (Collins, 2002) is performed. The hypothesis is parsed again with the new lexicon, and an update to the parameters $\mathbf{w}$ is made if the resulting parse does not have the correct logical form.

This algorithm differs from the approach in ZC05 in a couple of important respects. First, the ZC05 algorithm performed learning of the lexicon Λ at each iteration in a batch method, requiring a pass over the entire training set. The new algorithm is fully online, learning both Λ and $\mathbf{w}$ in an example-by-example fashion. This has important consequences for the efficiency of the algorithm. Second, the parameter estimation method in ZC05 was based on stochastic gradient descent on a log-likelihood objective function. The new algorithm makes use of perceptron

[8] In general, applications of the crossed composition rules can be lexically governed, as described in work on Multi-Modal CCG (Baldridge, 2002). In the future we would like to incorporate more fine-grained lexical distinctions of this type.

```
Inputs: Training examples $\{(x_i, z_i) : i = 1 \ldots n\}$ where
    each $x_i$ is a sentence, each $z_i$ is a logical form. An initial
    lexicon $\Lambda_0$. Number of training iterations, $T$.

Definitions: GENLEX$(x, z)$ takes as input a sentence $x$ and
    a logical form $z$ and returns a set of lexical items as described
    in section 2.4. GEN$(x; \Lambda)$ is the set of all parses
    for $x$ with lexicon $\Lambda$. GEN$(x, z; \Lambda)$ is the set of all parses
    for $x$ with lexicon $\Lambda$, which have logical form $z$. The
    function $\mathbf{f}(x, y)$ represents the features described in
    section 4.1. The function $L(y)$ maps a parse tree $y$ to its
    associated logical form.

Initialization: Set parameters $\mathbf{w}$ to initial values described in
    section 6.2. Set $\Lambda = \Lambda_0$.

Algorithm:
    • For $t = 1 \ldots T, i = 1 \ldots n$ :
    Step 1: (Check correctness)
        • Let $y^* = \arg\max_{y \in \text{GEN}(x_i;\Lambda)} \mathbf{w} \cdot \mathbf{f}(x_i, y)$ .
        • If $L(y^*) = z_i$, go to the next example.
    Step 2: (Lexical generation)
        • Set $\lambda = \Lambda \cup \text{GENLEX}(x_i, z_i)$ .
        • Let $y^* = \arg\max_{y \in \text{GEN}(x_i,z_i;\lambda)} \mathbf{w} \cdot \mathbf{f}(x_i, y)$ .
        • Define $\lambda_i$ to be the set of lexical entries in $y^*$.
        • Set lexicon to $\Lambda = \Lambda \cup \lambda_i$ .
    Step 3: (Update parameters)
        • Let $y' = \arg\max_{y \in \text{GEN}(x_i;\Lambda)} \mathbf{w} \cdot \mathbf{f}(x_i, y)$ .
        • If $L(y') \neq z_i$ :
            • Set $\mathbf{w} = \mathbf{w} + \mathbf{f}(x_i, y^*) - \mathbf{f}(x_i, y')$ .

Output: Lexicon $\Lambda$ together with parameters $\mathbf{w}$.
```

Figure 4: An online learning algorithm.

updates, which are simpler and cheaper to compute.

As in ZC05, the algorithm assumes an initial lexicon Λ_0 that contains two types of entries. First, we compile entries such as $Boston := NP : boston$ for entities such as cities, times and month-names that occur in the domain or underlying database. In practice it is easy to compile a list of these atomic entities. Second, the lexicon has entries for some function words such as wh-words, and determiners.[9]

5 Related Work

There has been a significant amount of previous work on learning to map sentences to underlying semantic representations. A wide variety of techniques have been considered including approaches based on machine translation techniques (Papineni et al., 1997; Ramaswamy and Kleindienst, 2000; Wong and Mooney, 2006), parsing techniques (Miller et al., 1996; Ge and Mooney, 2006), techniques that use inductive logic programming (Zelle and Mooney, 1996; Thompson and Mooney, 2002; Tang and Mooney, 2000; Kate et al., 2005), and ideas from string kernels and support vector machines (Kate and Mooney, 2006; Nguyen et al., 2006). In our experiments we compare to He and Young (2006) on the ATIS domain and Zettlemoyer and Collins (2005) on the Geo880 domain, because these systems currently achieve the best performance on these problems.

The approach of Zettlemoyer and Collins (2005) was presented in section 2.4. He and Young (2005) describe an algorithm that learns a probabilistic push-down automaton that models hierarchical dependencies but can still be trained on a data set that does not have full treebank-style annotations. This approach has been integrated with a speech recognizer and shown to be robust to recognition errors (He and Young, 2006).

There is also related work in the CCG literature. Clark and Curran (2003) present a method for learning the parameters of a log-linear CCG parsing model from fully annotated normal–form parse trees. Watkinson and Manandhar (1999) present an unsupervised approach for learning CCG lexicons that does not represent the semantics of the training sentences. Bos et al. (2004) present an algorithm that learns CCG lexicons with semantics but requires fully–specified CCG derivations in the training data. Bozsahin (1998) presents work on using CCG to model languages with free word order.

In addition, there is related work that focuses on modeling child language learning. Siskind (1996) presents an algorithm that learns word-to-meaning mappings from sentences that are paired with a set of possible meaning representations. Villavicencio (2001) describes an approach that learns a categorial grammar with syntactic and semantic information. Both of these approaches use sentences from child-directed speech, which differ significantly from the natural language interface queries we consider.

Finally, there is work on manually developing parsing techniques to improve robustness (Carbonell

[9] Our assumption is that these entries are likely to be domain independent, so it is simple enough to compile a list that can be reused in new domains. Another approach, which we may consider in the future, would be to annotate a small subset of the training examples with full CCG derivations, from which these frequently occurring entries could be learned.

and Hayes, 1983; Seneff, 1992). In contrast, our approach is integrated into a learning framework.

6 Experiments

The main focus of our experiments is on the ATIS travel planning domain. For development, we used 4978 sentences, split into a training set of 4500 examples, and a development set of 478 examples. For test, we used the ATIS NOV93 test set which contains 448 examples. To create the annotations, we created a script that maps the original SQL annotations provided with the data to lambda-calculus expressions.

He and Young (2006) previously reported results on the ATIS domain, using a learning approach which also takes sentences paired with semantic annotations as input. In their case, the semantic structures resemble context-free parses with semantic (as opposed to syntactic) non-terminal labels. In our experiments we have used the same split into training and test data as He and Young (2006), ensuring that our results are directly comparable. He and Young (2006) report *partial match* figures for their parser, based on precision and recall in recovering attribute-value pairs. (For example, the sentence *flights to Boston* would have a single attribute-value entry, namely $destination = Boston$.) It is simple for us to map from lambda-calculus expressions to attribute-value entries of this form; for example, the expression $to(x, Boston)$ would be mapped to $destination = Boston$. He and Young (2006) gave us their data and annotations, so we can directly compare results on the partial-match criterion. We also report accuracy for exact matches of lambda-calculus expressions, which is a stricter criterion.

In addition, we report results for the method on the Geo880 domain. This allows us to compare directly to the previous work of Zettlemoyer and Collins (2005), using the same split of the data into training and test sets of sizes 600 and 280 respectively. We use cross-validation of the training set, as opposed to a separate development set, for optimization of parameters.

6.1 Improving Recall

The simplest approach to the task is to train the parser and directly apply it to test sentences. In our experiments we will see that this produces results which have high precision, but somewhat lower recall, due to some test sentences failing to parse (usually due to words in the test set which were never observed in training data). A simple strategy to alleviate this problem is as follows. If the sentence fails to parse, we parse the sentence again, this time allowing parse moves which can delete words at some cost. The cost of this deletion operation is optimized on development data. This approach can significantly improve F-measure on the partial-match criterion in particular. We report results both with and without this second pass strategy.

6.2 Parameters in the Approach

The algorithm in figure 4 has a number of parameters, the set $\{T, \alpha, \beta, \gamma\}$, which we now describe. The values of these parameters were chosen to optimize the performance on development data. T is the number of passes over the training set, and was set to be 4. Each lexical entry in the initial lexicon Λ_0 has an associated feature which counts the number of times this entry is seen in a parse. The initial parameter value in $\mathbf{w}$ for all features of this form was chosen to be some value α. Each of the new CCG rules—the application, composition, crossed-composition, and type-raising rules described in section 3—has an associated parameter. We set all of these parameters to the same initial value β. Finally, when new lexical entries are added to Λ (in step 2 of the algorithm), their initial weight is set to some value γ. In practice, optimization on development data led to a positive value for α, and negative values for β and γ.

6.3 Results

Table 1 shows accuracy for the method by the exact-match criterion on the ATIS test set. The two pass strategy actually hurts F-measure in this case, although it does improve recall of the method.

Table 2 shows results under the partial-match criterion. The results for our approach are higher than those reported by He and Young (2006) even without the second, high-recall, strategy. With the two-pass strategy our method has more than halved the F-measure error rate, giving improvements from 90.3% F-measure to 95.9% F-measure.

Table 3 shows results on the Geo880 domain. The

	Precision	Recall	F1
Single-Pass Parsing	90.61	81.92	86.05
Two-Pass Parsing	85.75	84.6	85.16

Table 1: Exact-match accuracy on the ATIS test set.

	Precision	Recall	F1
Single-Pass Parsing	96.76	86.89	91.56
Two-Pass Parsing	95.11	96.71	95.9
He and Young (2006)	–	–	90.3

Table 2: Partial-credit accuracy on the ATIS test set.

new method gives improvements in performance both with and without the two pass strategy, showing that the new CCG combinators, and the new learning algorithm, give some improvement on even this domain. The improved performance comes from a slight drop in precision which is offset by a large increase in recall.

Table 4 shows ablation studies on the ATIS data, where we have selectively removed various aspects of the approach, to measure their impact on performance. It can be seen that accuracy is seriously degraded if the new CCG rules are removed, or if the features associated with these rules (which allow the model to penalize these rules) are removed.

Finally, we report results concerning the efficiency of the new online algorithm as compared to the ZC05 algorithm. We compared running times for the new algorithm, and the ZC05 algorithm, on the geography domain, with both methods making 4 passes over the training data. The new algorithm took less than 4 hours, compared to over 12 hours for the ZC05 algorithm. The main explanation for this improved performance is that on many training examples,[10] in step 1 of the new algorithm a correct parse is found, and the algorithm immediately moves on to the next example. Thus GENLEX is not required, and in particular parsing the example with the large set of entries generated by GENLEX is not required.

7 Discussion

We presented a new, online algorithm for learning a combinatory categorial grammar (CCG), together with parameters that define a log-linear parsing model. We showed that the use of non-standard CCG combinators is highly effective for parsing sentences with the types of phenomena seen in spontaneous, unedited natural language. The resulting system achieved significant accuracy improvements in both the ATIS and Geo880 domains.

	Precision	Recall	F1
Single-Pass Parsing	95.49	83.2	88.93
Two-Pass Parsing	91.63	86.07	88.76
ZC05	96.25	79.29	86.95

Table 3: Exact-match accuracy on the Geo880 test set.

	Precision	Recall	F1
Full Online Method	87.26	74.44	80.35
Without control features	70.33	42.45	52.95
Without relaxed word order	82.81	63.98	72.19
Without word insertion	77.31	56.94	65.58

Table 4: Exact-match accuracy on the ATIS development set for the full algorithm and restricted versions of it. The second row reports results of the approach without the features described in section 3 that control the use of the new combinators. The third row presents results without the combinators from section 3.1 that relax word order. The fourth row reports experiments without the type-raising combinators presented in section 3.2.

Acknowledgements

We would like to thank Yulan He and Steve Young for their help with obtaining the ATIS data set. We also acknowledge the support for this research. Luke Zettlemoyer was funded by a Microsoft graduate research fellowship and Michael Collins was supported by the National Science Foundation under grants 0347631 and DMS-0434222.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Cem Bozsahin. 1998. Deriving the predicate-argument structure for a free word order language. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*.

Jaime G. Carbonell and Philip J. Hayes. 1983. Recovery strategies for parsing extragrammatical language. *American Journal of Computational Linguistics*, 9.

Bob Carpenter. 1997. *Type-Logical Semantics*. The MIT Press.

Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[10] Measurements on the Geo880 domain showed that in the 4 iterations, 83.3% of all parses were successful at step 1.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

Michael Collins. 2004. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In Harry Bunt, John Carroll and Giorgio Satta, editors, *New Developments in Parsing Technology*. Kluwer.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: the atis-3 corpus. In *ARPA Human Language Technology Workshop*.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

Yulan He and Steve Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech and Language*.

Yulan He and Steve Young. 2006. Spoken language understanding using the hidden vector state model. *Speech Communication Special Issue on Spoken Language Understanding for Conversational Systems*.

Mark Johnson, Stuart Geman, Steven Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the Association for Computational Linguistics*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Scott Miller, David Stallard, Robert J. Bobrow, and Richard L. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the Association for Computational Linguistics*.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *Computational Linguistics and Intelligent Text Processing: Proceedings of the 8th International Conference*.

Le-Minh Nguyen, Akira Shimazu, and Xuan-Hieu Phan. 2006. Semantic parsing with structured SVM ensemble classification models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

K. A. Papineni, S. Roukos, and T. R. Ward. 1997. Feature-based language understanding. In *Proceedings of European Conference on Speech Communication and Technology*.

Ganesh N. Ramaswamy and Jan Kleindienst. 2000. Hierarchical feature-based translation for scalable natural language understanding. In *Proceedings of 6th International Conference on Spoken Language Processing*.

Adwait Ratnaparkhi, Salim Roukos, and R. Todd Ward. 1994. A maximum entropy model for parsing. In *Proceedings of the International Conference on Spoken Language Processing*.

Stephanie Seneff. 1992. Robust parsing for spoken language systems. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*.

Jeffrey M. Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(2-3).

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Cynthia A. Thompson and Raymond J. Mooney. 2002. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, 18.

Aline Villavicencio. 2001. *The acquisition of a unification-based generalised categorial grammar*. Ph.D. thesis, University of Cambridge.

Stephen Watkinson and Suresh Manandhar. 1999. Unsupervised lexical learning with categorial grammars using the LLL corpus. In *Proceedings of the 1st Workshop on Learning Language in Logic*.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 14th National Conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.

The Infinite PCFG using Hierarchical Dirichlet Processes

Percy Liang **Slav Petrov** **Michael I. Jordan** **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{pliang, petrov, jordan, klein}@cs.berkeley.edu

Abstract

We present a nonparametric Bayesian model of tree structures based on the hierarchical Dirichlet process (HDP). Our HDP-PCFG model allows the complexity of the grammar to grow as more training data is available. In addition to presenting a fully Bayesian model for the PCFG, we also develop an efficient variational inference procedure. On synthetic data, we recover the correct grammar without having to specify its complexity in advance. We also show that our techniques can be applied to full-scale parsing applications by demonstrating its effectiveness in learning state-split grammars.

1 Introduction

Probabilistic context-free grammars (PCFGs) have been a core modeling technique for many aspects of linguistic structure, particularly syntactic phrase structure in treebank parsing (Charniak, 1996; Collins, 1999). An important question when learning PCFGs is how many grammar symbols to allocate to the learning algorithm based on the amount of available data.

The question of "how many clusters (symbols)?" has been tackled in the Bayesian nonparametrics literature via Dirichlet process (DP) mixture models (Antoniak, 1974). DP mixture models have since been extended to hierarchical Dirichlet processes (HDPs) and HDP-HMMs (Teh et al., 2006; Beal et al., 2002) and applied to many different types of clustering/induction problems in NLP (Johnson et al., 2006; Goldwater et al., 2006).

In this paper, we present the *hierarchical Dirichlet process PCFG* (HDP-PCFG). a nonparametric Bayesian model of syntactic tree structures based on Dirichlet processes. Specifically, an HDP-PCFG is defined to have an infinite number of symbols; the Dirichlet process (DP) prior penalizes the use of more symbols than are supported by the training data. Note that "nonparametric" does not mean "no parameters"; rather, it means that the effective number of parameters can grow adaptively as the amount of data increases, which is a desirable property of a learning algorithm.

As models increase in complexity, so does the uncertainty over parameter estimates. In this regime, point estimates are unreliable since they do not take into account the fact that there are different amounts of uncertainty in the various components of the parameters. The HDP-PCFG is a Bayesian model which naturally handles this uncertainty. We present an efficient variational inference algorithm for the HDP-PCFG based on a structured mean-field approximation of the true posterior over parameters. The algorithm is similar in form to EM and thus inherits its simplicity, modularity, and efficiency. Unlike EM, however, the algorithm is able to take the uncertainty of parameters into account and thus incorporate the DP prior.

Finally, we develop an extension of the HDP-PCFG for grammar refinement (HDP-PCFG-GR). Since treebanks generally consist of coarsely-labeled context-free tree structures, the maximum-likelihood treebank grammar is typically a poor model as it makes overly strong independence assumptions. As a result, many generative approaches to parsing construct refinements of the treebank grammar which are more suitable for the modeling task. Lexical methods split each pre-terminal symbol into many subsymbols, one for each word, and then focus on smoothing sparse lexical statis-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 688–697, Prague, June 2007. ©2007 Association for Computational Linguistics

tics (Collins, 1999; Charniak, 2000). Unlexicalized methods refine the grammar in a more conservative fashion, splitting each non-terminal or pre-terminal symbol into a much smaller number of subsymbols (Klein and Manning, 2003; Matsuzaki et al., 2005; Petrov et al., 2006). We apply our HDP-PCFG-GR model to automatically learn the number of subsymbols for each symbol.

2 Models based on Dirichlet processes

At the heart of the HDP-PCFG is the Dirichlet process (DP) mixture model (Antoniak, 1974), which is the nonparametric Bayesian counterpart to the classical finite mixture model. In order to build up an understanding of the HDP-PCFG, we first review the Bayesian treatment of the finite mixture model (Section 2.1). We then consider the DP mixture model (Section 2.2) and use it as a building block for developing nonparametric structured versions of the HMM (Section 2.3) and PCFG (Section 2.4). Our presentation highlights the similarities between these models so that each step along this progression reflects only the key differences.

2.1 Bayesian finite mixture model

We begin by describing the Bayesian finite mixture model to establish basic notation that will carry over the more complex models we consider later.

```
Bayesian finite mixture model
$\boldsymbol{\beta} \sim$ Dirichlet$(\alpha, \dots, \alpha)$      [draw component probabilities]
For each component $z \in \{1, \dots, K\}$:
  $\phi_z \sim G_0$                              [draw component parameters]

For each data point $i \in \{1, \dots, n\}$:
  $z_i \sim$ Multinomial$(\boldsymbol{\beta})$                     [choose component]
  $x_i \sim F(\cdot; \phi_{z_i})$                          [generate data point]
```

The model has K components whose prior distribution is specified by $\boldsymbol{\beta} = (\beta_1, \dots, \beta_K)$. The Dirichlet hyperparameter α controls how uniform this distribution is: as α increases, it becomes increasingly likely that the components have equal probability. For each mixture component $z \in \{1, \dots, K\}$, the parameters of the component ϕ_z are drawn from some prior G_0. Given the model parameters $(\boldsymbol{\beta}, \boldsymbol{\phi})$, the data points are generated i.i.d. by first choosing a component and then generating from a data model F parameterized by that component.

In document clustering, for example, each data point x_i is a document represented by its term-frequency vector. Each component (cluster) z has multinomial parameters ϕ_z which specifies a distribution $F(\cdot; \phi_z)$ over words. It is customary to use a conjugate Dirichlet prior $G_0 =$ Dirichlet$(\alpha', \dots, \alpha')$ over the multinomial parameters, which can be interpreted as adding $\alpha' - 1$ pseudocounts for each word.

2.2 DP mixture model

We now consider the extension of the Bayesian finite mixture model to a nonparametric Bayesian mixture model based on the Dirichlet process. We focus on the *stick-breaking representation* (Sethuraman, 1994) of the Dirichlet process instead of the stochastic process definition (Ferguson, 1973) or the Chinese restaurant process (Pitman, 2002). The stick-breaking representation captures the DP prior most explicitly and allows us to extend the finite mixture model with minimal changes. Later, it will enable us to readily define structured models in a form similar to their classical versions. Furthermore, an efficient variational inference algorithm can be developed in this representation (Section 2.6).

The key difference between the Bayesian finite mixture model and the DP mixture model is that the latter has a countably infinite number of mixture components while the former has a predefined K. Note that if we have an infinite number of mixture components, it no longer makes sense to consider a symmetric prior over the component probabilities; the prior over component probabilities must decay in some way. The stick-breaking distribution achieves this as follows. We write $\boldsymbol{\beta} \sim$ GEM(α) to mean that $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots)$ is distributed according to the stick-breaking distribution. Here, the concentration parameter α controls the number of effective components. To draw $\boldsymbol{\beta} \sim$ GEM(α), we first generate a countably infinite collection of stick-breaking *proportions* $u_1, u_2, \dots$, where each $u_z \sim$ Beta$(1, \alpha)$. The stick-breaking weights $\boldsymbol{\beta}$ are then defined in terms of the stick proportions:

$$\beta_z = u_z \prod_{z' < z} (1 - u_{z'}). \tag{1}$$

The procedure for generating $\boldsymbol{\beta}$ can be viewed as iteratively breaking off remaining portions of a unit-

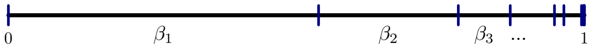


Figure 1: A sample $\boldsymbol{\beta} \sim \text{GEM}(1)$.

length stick (Figure 1). The component probabilities $\{\beta_z\}$ will decay exponentially in expectation, but there is always some probability of getting a smaller component before a larger one. The parameter α determines the decay of these probabilities: a larger α implies a slower decay and thus more components.

Given the component probabilities, the rest of the DP mixture model is identical to the finite mixture model:

```
DP mixture model
$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$                      [draw component probabilities]
For each component $z \in \{1, 2, \dots\}$:
    $\phi_z \sim G_0$                             [draw component parameters]

For each data point $i \in \{1, \dots, n\}$:
    $z_i \sim \text{Multinomial}(\boldsymbol{\beta})$                     [choose component]
    $x_i \sim F(\cdot; \phi_{z_i})$                        [generate data point $x_n$]
```

2.3 HDP-HMM

The next stop on the way to the HDP-PCFG is the HDP hidden Markov model (HDP-HMM) (Beal et al., 2002; Teh et al., 2006). An HMM consists of a set of hidden states, where each state can be thought of as a mixture component. The parameters of the mixture component are the emission and transition parameters. The main aspect that distinguishes it from a flat finite mixture model is that the transition parameters themselves must specify a distribution over next states. Hence, we have not just one top-level mixture model over states, but also a collection of mixture models, one for each state.

In developing a nonparametric version of the HMM in which the number of states is infinite, we need to ensure that the transition mixture models of each state share a common inventory of possible next states. We can achieve this by tying these mixture models together using the hierarchical Dirichlet process (HDP) (Teh et al., 2006). The stick-breaking representation of an HDP is defined as follows: first, the top-level stick-breaking weights $\boldsymbol{\beta}$ are drawn according to the stick-breaking prior as before. Then, a new set of stick-breaking weights $\boldsymbol{\beta}'$ are generated according based on $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}' \sim \text{DP}(\alpha', \boldsymbol{\beta}), \tag{2}$$

where the distribution of DP can be characterized in terms of the following finite partition property: for all partitions of the positive integers into sets $A_1, \dots, A_m$,

$$\begin{aligned}(\boldsymbol{\beta}'(A_1), \dots, \boldsymbol{\beta}'(A_m)) \\ \sim \text{Dirichlet}\Big(\alpha'\boldsymbol{\beta}(A_1), \dots, \alpha'\boldsymbol{\beta}(A_m)\Big),\end{aligned} \tag{3}$$

where $\boldsymbol{\beta}(A) = \sum_{k \in A} \beta_k$.[1] The resulting $\boldsymbol{\beta}'$ is another distribution over the positive integers whose similarity to $\boldsymbol{\beta}$ is controlled by a concentration parameter α'.

```
HDP-HMM
$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$                    [draw top-level state weights]
For each state $z \in \{1, 2, \dots\}$:
    $\phi_z^E \sim \text{Dirichlet}(\gamma)$              [draw emission parameters]
    $\phi_z^T \sim \text{DP}(\alpha', \beta)$             [draw transition parameters]

For each time step $i \in \{1, \dots, n\}$:
    $x_i \sim F(\cdot; \phi_{z_i}^E)$                   [emit current observation]
    $z_{i+1} \sim \text{Multinomial}(\phi_{z_i}^T)$                [choose next state]
```

Each state z is associated with emission parameters ϕ_z^E. In addition, each z is also associated with transition parameters ϕ_z^T, which specify a distribution over next states. These transition parameters are drawn from a DP centered on the top-level stick-breaking weights $\boldsymbol{\beta}$ according to Equations (2) and (3). Assume that z_1 is always fixed to a special START state, so we do not need to generate it.

2.4 HDP-PCFG

We now present the HDP-PCFG, which is the focus of this paper. For simplicity, we consider Chomsky normal form (CNF) grammars, which has two types of rules: emissions and binary productions. We consider each grammar symbol as a mixture component whose parameters are the rule probabilities for that symbol. In general, we do not know the appropriate number of grammar symbols, so our strategy is to let the number of grammar symbols be infinite and place a DP prior over grammar symbols.

[1] Note that this property is a specific instance of the general stochastic process definition of Dirichlet processes.

```
HDP-PCFG
$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$                                  [draw top-level symbol weights]
For each grammar symbol $z \in \{1, 2, \dots\}$:
  $\phi^T_z \sim \text{Dirichlet}(\alpha^T)$                         [draw rule type parameters]
  $\phi^E_z \sim \text{Dirichlet}(\alpha^E)$                         [draw emission parameters]
  $\phi^B_z \sim \text{DP}(\alpha^B, \boldsymbol{\beta}\boldsymbol{\beta}^T)$            [draw binary production parameters]

For each node $i$ in the parse tree:
  $t_i \sim \text{Multinomial}(\phi^T_{z_i})$                               [choose rule type]
  If $t_i = \text{EMISSION}$:
    $x_i \sim \text{Multinomial}(\phi^E_{z_i})$                          [emit terminal symbol]
  If $t_i = \text{BINARY-PRODUCTION}$:
    $(z_{L(i)}, z_{R(i)}) \sim \text{Multinomial}(\phi^B_{z_i})$    [generate children symbols]
```

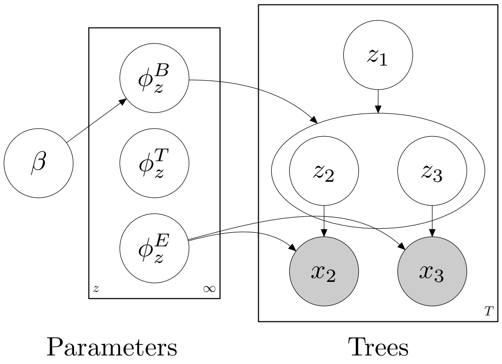


Figure 2: The definition and graphical model of the HDP-PCFG. Since parse trees have unknown structure, there is no convenient way of representing them in the visual language of traditional graphical models. Instead, we show a simple fixed example tree. Node 1 has two children, 2 and 3, each of which has one observed terminal child. We use $L(i)$ and $R(i)$ to denote the left and right children of node i.

In the HMM, the transition parameters of a state specify a distribution over single next states; similarly, the binary production parameters of a grammar symbol must specify a distribution over pairs of grammar symbols for its children. We adapt the HDP machinery to tie these binary production distributions together. The key difference is that now we must tie distributions over pairs of grammar symbols together via distributions over single grammar symbols.

Another difference is that in the HMM, at each time step, both a transition and a emission are made, whereas in the PCFG *either* a binary production or an emission is chosen. Therefore, each grammar symbol must also have a distribution over the type of rule to apply. In a CNF PCFG, there are only two types of rules, but this can be easily generalized to include unary productions, which we use for our parsing experiments.

To summarize, the parameters of each grammar symbol z consists of (1) a distribution over a finite number of rule types ϕ^T_z, (2) an emission distribution ϕ^E_z over terminal symbols, and (3) a binary production distribution ϕ^B_z over pairs of children grammar symbols. Figure 2 describes the model in detail.

Figure 3 shows the generation of the binary production distributions ϕ^B_z. We draw ϕ^B_z from a DP centered on $\boldsymbol{\beta}\boldsymbol{\beta}^T$, which is the product distribution over pairs of symbols. The result is a doubly-infinite matrix where most of the probability mass is concentrated in the upper left, just like the top-level distribution $\boldsymbol{\beta}\boldsymbol{\beta}^T$.

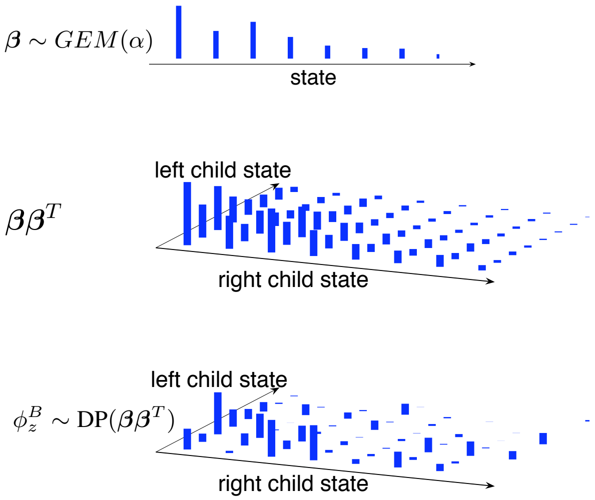


Figure 3: The generation of binary production probabilities given the top-level symbol probabilities $\boldsymbol{\beta}$. First, $\boldsymbol{\beta}$ is drawn from the stick-breaking prior, as in any DP-based model (a). Next, the outer-product $\boldsymbol{\beta}\boldsymbol{\beta}^T$ is formed, resulting in a doubly-infinite matrix matrix (b). We use this as the base distribution for generating the binary production distribution from a DP centered on $\boldsymbol{\beta}\boldsymbol{\beta}^T$ (c).

Note that we have replaced the general

G_0 and $F(\phi^E_{z_i})$ pair with Dirichlet(α^E) and Multinomial($\phi^E_{z_i}$) to specialize to natural language, but there is no difficulty in working with parse trees with arbitrary non-multinomial observations or more sophisticated word models.

In many natural language applications, there is a hard distinction between pre-terminal symbols (those that only emit a word) and non-terminal symbols (those that only rewrite as two non-terminal or pre-terminal symbols). This can be accomplished by letting $\alpha^T = (0, 0)$, which forces a draw ϕ^T_z to assign probability 1 to one rule type.

An alternative definition of an HDP-PCFG would be as follows: for each symbol z, draw a distribution over left child symbols $l_z \sim \mathrm{DP}(\boldsymbol{\beta})$ and an independent distribution over right child symbols $r_z \sim \mathrm{DP}(\boldsymbol{\beta})$. Then define the binary production distribution as their cross-product $\phi^B_z = l_z r_z^T$. This also yields a distribution over symbol pairs and hence defines a different type of nonparametric PCFG. This model is simpler and does not require any additional machinery beyond the HDP-HMM. However, the modeling assumptions imposed by this alternative are unappealing as they assume the left child and right child are independent given the parent, which is certainly not the case in natural language.

2.5 HDP-PCFG for grammar refinement

An important motivation for the HDP-PCFG is that of refining an existing treebank grammar to alleviate unrealistic independence assumptions and to improve parsing accuracy. In this scenario, the set of symbols is known, but we do not know how many subsymbols to allocate per symbol. We introduce the HDP-PCFG for grammar refinement (HDP-PCFG-GR), an extension of the HDP-PCFG, for this task.

The essential difference is that now we have a collection of HDP-PCFG models for each symbol $s \in S$, each one operating at the subsymbol level. While these HDP-PCFGs are independent in the prior, they are coupled through their interactions in the parse trees. For completeness, we have also included unary productions, which are essentially the PCFG counterpart of transitions in HMMs. Finally, since each node i in the parse tree involves a symbol-subsymbol pair (s_i, z_i), each subsymbol needs to specify a distribution over both child symbols and subsymbols. The former can be handled through a finite Dirichlet distribution since all symbols are known and observed, but the latter must be handled with the Dirichlet process machinery, since the number of subsymbols is unknown.

```
HDP-PCFG for grammar refinement (HDP-PCFG-GR)

For each symbol $s \in S$:
  $\boldsymbol{\beta}_s \sim \mathrm{GEM}(\alpha)$                          [draw subsymbol weights]
  For each subsymbol $z \in \{1, 2, \dots\}$:
    $\phi^T_{sz} \sim \mathrm{Dirichlet}(\alpha^T)$               [draw rule type parameters]
    $\phi^E_{sz} \sim \mathrm{Dirichlet}(\alpha^E(s))$            [draw emission parameters]
    $\phi^u_{sz} \sim \mathrm{Dirichlet}(\alpha^u)$               [unary symbol productions]
    $\phi^b_{sz} \sim \mathrm{Dirichlet}(\alpha^b)$              [binary symbol productions]
    For each child symbol $s' \in S$:
      $\phi^U_{szs'} \sim \mathrm{DP}(\alpha^U, \boldsymbol{\beta}_{s'})$            [unary subsymbol prod.]
    For each pair of children symbols $(s', s'') \in S \times S$:
      $\phi^B_{szs's''} \sim \mathrm{DP}(\alpha^B, \boldsymbol{\beta}_{s'}\boldsymbol{\beta}^T_{s''})$        [binary subsymbol]

For each node $i$ in the parse tree:
  $t_i \sim \mathrm{Multinomial}(\phi^T_{s_i z_i})$                        [choose rule type]
  If $t_i = $ EMISSION:
    $x_i \sim \mathrm{Multinomial}(\phi^E_{s_i z_i})$              [emit terminal symbol]
  If $t_i = $ UNARY-PRODUCTION:
    $s_{L(i)} \sim \mathrm{Multinomial}(\phi^u_{s_i z_i})$        [generate child symbol]
    $z_{L(i)} \sim \mathrm{Multinomial}(\phi^U_{s_i z_i s_{L(i)}})$          [child subsymbol]
  If $t_i = $ BINARY-PRODUCTION:
    $(s_{L(i)}, s_{R(i)}) \sim \mathrm{Mult}(\phi_{s_i z_i})$              [children symbols]
    $(z_{L(i)}, z_{R(i)}) \sim \mathrm{Mult}(\phi^B_{s_i z_i s_{L(i)} s_{R(i)}})$        [subsymbols]
```

2.6 Variational inference

We present an inference algorithm for the HDP-PCFG model described in Section 2.4, which can also be adapted to the HDP-PCFG-GR model with a bit more bookkeeping. Most previous inference algorithms for DP-based models involve sampling (Escobar and West, 1995; Teh et al., 2006). However, we chose to use variational inference (Blei and Jordan, 2005), which provides a fast deterministic alternative to sampling, hence avoiding issues of diagnosing convergence and aggregating samples. Furthermore, our variational inference algorithm establishes a strong link with past work on PCFG refinement and induction, which has traditionally employed the EM algorithm.

In EM, the E-step involves a dynamic program that exploits the Markov structure of the parse tree, and the M-step involves computing ratios based on expected counts extracted from the E-step. Our variational algorithm resembles the EM algorithm in form, but the ratios in the M-step are replaced with weights that reflect the uncertainty in parameter es-

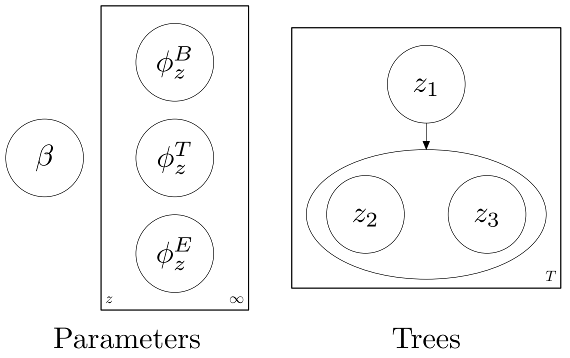


Figure 4: We approximate the true posterior p over parameters $\boldsymbol{\theta}$ and latent parse trees $\mathbf{z}$ using a structured mean-field distribution q, in which the distribution over parameters are completely factorized but the distribution over parse trees is unconstrained.

timates. Because of this procedural similarity, our method is able to exploit the desirable properties of EM such as simplicity, modularity, and efficiency.

2.7 Structured mean-field approximation

We denote parameters of the HDP-PCFG as $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\phi})$, where $\boldsymbol{\beta}$ denotes the top-level symbol probabilities and $\boldsymbol{\phi}$ denotes the rule probabilities. The hidden variables of the model are the training parse trees $\mathbf{z}$. We denote the observed sentences as $\mathbf{x}$.

The goal of Bayesian inference is to compute the posterior distribution $p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x})$. The central idea behind variational inference is to approximate this intractable posterior with a tractable approximation. In particular, we want to find the best distribution q^* as defined by

$$q^* \stackrel{\text{def}}{=} \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}(q(\boldsymbol{\theta}, \mathbf{z}) || p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x})), \quad (4)$$

where $\mathcal{Q}$ is a tractable subset of distributions. We use a *structured mean-field approximation*, meaning that we only consider distributions that factorize as follows (Figure 4):

$$\mathcal{Q} \stackrel{\text{def}}{=} \Big\{ q(\mathbf{z}) q(\boldsymbol{\beta}) \prod_{z=1}^{K} q(\phi_z^T) q(\phi_z^E) q(\phi_z^B) \Big\}. \quad (5)$$

We further restrict $q(\phi_z^T), q(\phi_z^E), q(\phi_z^B)$ to be Dirichlet distributions, but allow $q(\mathbf{z})$ to be any multinomial distribution. We constrain $q(\boldsymbol{\beta})$ to be a degenerate distribution truncated at K; i.e., $\beta_z = 0$ for $z > K$. While the posterior grammar does have an infinite number of symbols, the exponential decay of the DP prior ensures that most of the probability mass is contained in the first few symbols (Ishwaran and James, 2001).[2] While our variational approximation q is truncated, the actual PCFG model is not. As K increases, our approximation improves.

2.8 Coordinate-wise ascent

The optimization problem defined by Equation (4) is intractable and nonconvex, but we can use a simple coordinate-ascent algorithm that iteratively optimizes each factor of q in turn while holding the others fixed. The algorithm turns out to be similar in form to EM for an ordinary PCFG: optimizing $q(\mathbf{z})$ is the analogue of the E-step, and optimizing $q(\boldsymbol{\phi})$ is the analogue of the M-step; however, optimizing $q(\boldsymbol{\beta})$ has no analogue in EM. We summarize each of these updates below (see (Liang et al., 2007) for complete derivations).

Parse trees $q(\mathbf{z})$: The distribution over parse trees $q(\mathbf{z})$ can be summarized by the expected sufficient statistics (rule counts), which we denote as $C(z \to z_l\, z_r)$ for binary productions and $C(z \to x)$ for emissions. We can compute these expected counts using dynamic programming as in the E-step of EM.

While the classical E-step uses the current rule probabilities $\boldsymbol{\phi}$, our mean-field approximation involves an entire distribution $q(\boldsymbol{\phi})$. Fortunately, we can still handle this case by replacing each rule probability with a weight that summarizes the uncertainty over the rule probability as represented by q. We define this weight in the sequel.

It is a common perception that Bayesian inference is slow because one needs to compute integrals. Our mean-field inference algorithm is a counterexample: because we can represent uncertainty over rule probabilities with single numbers, much of the existing PCFG machinery based on EM can be modularly imported into the Bayesian framework.

Rule probabilities $q(\boldsymbol{\phi})$: For an ordinary PCFG, the M-step simply involves taking ratios of expected

[2] In particular, the variational distance between the stick-breaking distribution and the truncated version decreases exponentially as the truncation level K increases.

counts:

$$\phi_z^B(z_l, z_r) = \frac{C(z \to z_l\, z_r)}{C(z \to *\,*)}. \tag{6}$$

For the variational HDP-PCFG, the optimal $q(\boldsymbol{\phi})$ is given by the standard posterior update for Dirichlet distributions:[3]

$$q(\phi_z^B) = \mathrm{Dirichlet}(\phi_z^B; \alpha^B \boldsymbol{\beta}\boldsymbol{\beta}^T + \vec{C}(z)), \tag{7}$$

where $\vec{C}(z)$ is the matrix of counts of rules with left-hand side z. These distributions can then be summarized with *multinomial weights* which are the only necessary quantities for updating $q(\mathbf{z})$ in the next iteration:

$$\begin{aligned} W_z^B(z_l, z_r) &\overset{\text{def}}{=} \exp \mathbb{E}_q[\log \boldsymbol{\phi}_z^B(z_l, z_r)] & (8)\\ &= \frac{e^{\Psi(C(z \to z_l\, z_r) + \alpha^B \beta_{z_l} \beta_{z_r})}}{e^{\Psi(C(z \to *\,*) + \alpha^B)}}, & (9) \end{aligned}$$

where $\Psi(\cdot)$ is the digamma function. The emission parameters can be defined similarly. Inspection of Equations (6) and (9) reveals that the only difference between the maximum likelihood and the mean-field update is that the latter applies the $\exp(\Psi(\cdot))$ function to the counts (Figure 5).

When the truncation K is large, $\alpha^B \beta_{z_l} \beta_{z_r}$ is near 0 for most right-hand sides (z_l, z_r), so $\exp(\Psi(\cdot))$ has the effect of downweighting counts. Since this subtraction affects large counts more than small counts, there is a rich-get-richer effect: rules that have already have large counts will be preferred.

Specifically, consider a set of rules with the same left-hand side. The weights for all these rules only differ in the numerator (Equation (9)), so applying $\exp(\Psi(\cdot))$ creates a *local preference* for right-hand sides with larger counts. Also note that the rule weights are not normalized; they always sum to at most one and are equal to one exactly when $q(\boldsymbol{\phi})$ is degenerate. This lack of normalization gives an extra degree of freedom not present in maximum likelihood estimation: it creates a *global preference* for left-hand sides that have larger total counts.

Top-level symbol probabilities $q(\boldsymbol{\beta})$**:** Recall that we restrict $q(\boldsymbol{\beta}) = \delta_{\boldsymbol{\beta}^*}(\boldsymbol{\beta})$, so optimizing $\boldsymbol{\beta}$ is equivalent to finding a single best $\boldsymbol{\beta}^*$. Unlike $q(\boldsymbol{\phi})$ and $q(\mathbf{z})$, there is no closed form expression for the optimal $\boldsymbol{\beta}^*$, and the objective function (Equation (4)) is not convex in $\boldsymbol{\beta}^*$. Nonetheless, we can apply a standard gradient projection method (Bertsekas, 1999) to improve $\boldsymbol{\beta}^*$ to a local maxima.

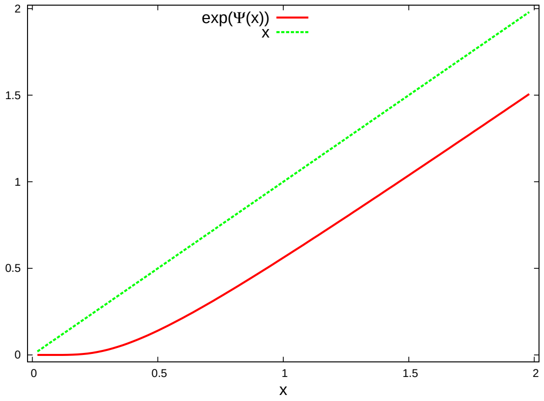


Figure 5: The $\exp(\Psi(\cdot))$ function, which is used in computing the multinomial weights for mean-field inference. It has the effect of reducing a larger fraction of small counts than large counts.

The part of the objective function in Equation (4) that depends on $\boldsymbol{\beta}^*$ is as follows:

$$\begin{aligned} L(\boldsymbol{\beta}^*) &= \log \mathrm{GEM}(\boldsymbol{\beta}^*; \alpha) + \qquad (10)\\ &\sum_{z=1}^{K} \mathbb{E}_q[\log \mathrm{Dirichlet}(\phi_z^B; \alpha^B \boldsymbol{\beta}^* \boldsymbol{\beta}^{*T})] \end{aligned}$$

See Liang et al. (2007) for the derivation of the gradient. In practice, this optimization has very little effect on performance. We suspect that this is because the objective function is dominated by $p(\mathbf{x} \mid \mathbf{z})$ and $p(\mathbf{z} \mid \boldsymbol{\phi})$, while the contribution of $p(\boldsymbol{\phi} \mid \boldsymbol{\beta})$ is minor.

3 Experiments

We now present an empirical evaluation of the HDP-PCFG(-GR) model and variational inference techniques. We first give an illustrative example of the ability of the HDP-PCFG to recover a known grammar and then present the results of experiments on large-scale treebank parsing.

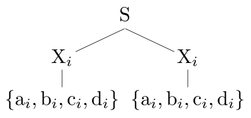
3.1 Recovering a synthetic grammar

In this section, we show that the HDP-PCFG-GR can recover a simple grammar while a standard

[3] Because we have truncated the top-level symbol weights, the DP prior on ϕ_z^B reduces to a finite Dirichlet distribution.

S $\rightarrow$ X_1X_1 | X_2X_2 | X_3X_3 | X_4X_4
$X_1 \rightarrow a_1$ | b_1 | c_1 | d_1
$X_2 \rightarrow a_2$ | b_2 | c_2 | d_2
$X_3 \rightarrow a_3$ | b_3 | c_3 | d_3
$X_4 \rightarrow a_4$ | b_4 | c_4 | d_4

(a)



(b)

Figure 6: (a) A synthetic grammar with a uniform distribution over rules. (b) The grammar generates trees of the form shown on the right.

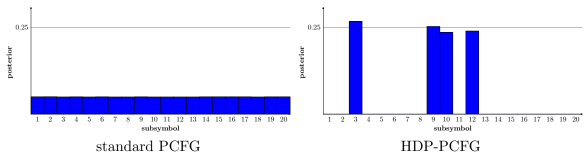


Figure 7: The posteriors over the subsymbols of the standard PCFG is roughly uniform, whereas the posteriors of the HDP-PCFG is concentrated on four subsymbols, which is the true number of symbols in the grammar.

PCFG fails to do so because it has no built-in control over grammar complexity. From the grammar in Figure 6, we generated 2000 trees. The two terminal symbols always have the same subscript, but we collapsed X_i to X in the training data. We trained the HDP-PCFG-GR, with truncation $K = 20$, for both S and X for 100 iterations. We set all hyperparameters to 1.

Figure 7 shows that the HDP-PCFG-GR recovers the original grammar, which contains only 4 subsymbols, leaving the other 16 subsymbols unused. The standard PCFG allocates all the subsymbols to fit the exact co-occurrence statistics of left and right terminals.

Recall that a rule weight, as defined in Equation (9), is analogous to a rule probability for standard PCFGs. We say a rule is *effective* if its weight is at least 10^{-6} and its left hand-side has posterior is also at least 10^{-6}. In general, rules with weight smaller than 10^{-6} can be safely pruned without affect parsing accuracy. The standard PCFG uses all 20 subsymbols of both S and X to explain the data, resulting in 8320 effective rules; in contrast, the HDP-PCFG uses only 4 subsymbols for X and 1 for S, resulting in only 68 effective rules. If the threshold is relaxed from 10^{-6} to 10^{-3}, then only 20 rules are effective, which corresponds exactly to the true grammar.

3.2 Parsing the Penn Treebank

In this section, we show that our variational HDP-PCFG can scale up to real-world data sets. We ran experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank. We trained on sections 2–21, used section 24 for tuning hyperparameters, and tested on section 22.

We binarize the trees in the treebank as follows: for each non-terminal node with symbol X, we introduce a right-branching cascade of new nodes with symbol $\overline{X}$. The end result is that each node has at most two children. To cope with unknown words, we replace any word appearing fewer than 5 times in the training set with one of 50 unknown word tokens derived from 10 word-form features.

Our goal is to learn a refined grammar, where each symbol in the training set is split into K subsymbols. We compare an ordinary PCFG estimated with maximum likelihood (Matsuzaki et al., 2005) and the HDP-PCFG estimated using the variational inference algorithm described in Section 2.6.

To parse new sentences with a grammar, we compute the posterior distribution over rules at each span and extract the tree with the maximum expected correct number of rules (Petrov and Klein, 2007).

3.2.1 Hyperparameters

There are six hyperparameters in the HDP-PCFG-GR model, which we set in the following manner: $\alpha = 1$, $\alpha^T = 1$ (uniform distribution over unaries versus binaries), $\alpha^E = 1$ (uniform distribution over terminal words), $\alpha^u(s) = \alpha^b(s) = \frac{1}{N(s)}$, where $N(s)$ is the number of different unary (binary) right-hand sides of rules with left-hand side s in the treebank grammar. The two most important hyperparameters are α^U and α^B, which govern the sparsity of the right-hand side for unary and binary rules. We set $\alpha^U = \alpha^B$ although more performance could probably be gained by tuning these individually. It turns out that there is not a single α^B that works for all truncation levels, as shown in Table 1.

If the top-level distribution $\boldsymbol{\beta}$ is uniform, the value of α^B corresponding to a uniform prior over pairs of children subsymbols is K^2. Interestingly, the optimal α^B appears to be superlinear but subquadratic

truncation K	2	4	8	12	16	20
best α^B	16	12	20	28	48	80
uniform α^B	4	16	64	144	256	400

Table 1: For each truncation level, we report the α^B that yielded the highest F_1 score on the development set.

K	PCFG		PCFG (smoothed)		HDP-PCFG	
	F_1	Size	F_1	Size	F_1	Size
1	60.47	2558	60.36	2597	60.5	2557
2	69.53	3788	69.38	4614	71.08	4264
4	75.98	3141	77.11	12436	77.17	9710
8	74.32	4262	79.26	120598	79.15	50629
12	70.99	7297	78.8	160403	78.94	86386
16	66.99	19616	79.2	261444	78.24	131377
20	64.44	27593	79.27	369699	77.81	202767

Table 2: Shows development F_1 and grammar sizes (the number of effective rules) as we increase the truncation K.

in K. We used these values of α^B in the following experiments.

3.2.2 Results

The regime in which Bayesian inference is most important is when training data is scarce relative to the complexity of the model. We train on just section 2 of the Penn Treebank. Table 2 shows how the HDP-PCFG-GR can produce compact grammars that guard against overfitting. Without smoothing, ordinary PCFGs trained using EM improve as K increases but start to overfit around $K = 4$. Simple add-1.01 smoothing prevents overfitting but at the cost of a sharp increase in grammar sizes. The HDP-PCFG obtains comparable performance with a much smaller number of rules.

We also trained on sections 2–21 to demonstrate that our methods can scale up and achieve broadly comparable results to existing state-of-the-art parsers. When using a truncation level of $K = 16$, the standard PCFG with smoothing obtains an F_1 score of 88.36 using 706157 effective rules while the HDP-PCFG-GR obtains an F_1 score of 87.08 using 428375 effective rules. We expect to see greater benefits from the HDP-PCFG with a larger truncation level.

4 Related work

The question of how to select the appropriate grammar complexity has been studied in earlier work. It is well known that more complex models necessarily have higher likelihood and thus a penalty must be imposed for more complex grammars. Examples of such penalized likelihood procedures include Stolcke and Omohundro (1994), which used an asymptotic Bayesian model selection criterion and Petrov et al. (2006), which used a split-merge algorithm which procedurally determines when to switch between grammars of various complexities. These techniques are model selection techniques that use heuristics to choose among competing statistical models; in contrast, the HDP-PCFG relies on the Bayesian formalism to provide implicit control over model complexity within the framework of a single probabilistic model.

Johnson et al. (2006) also explored nonparametric grammars, but they do not give an inference algorithm for recursive grammars, e.g., grammars including rules of the form $A \rightarrow BC$ and $B \rightarrow DA$. Recursion is a crucial aspect of PCFGs and our inference algorithm does handle it. Finkel et al. (2007) independently developed another nonparametric model of grammars. Though their model is also based on hierarchical Dirichlet processes and is similar to ours, they present a different inference algorithm which is based on sampling. Kurihara and Sato (2004) and Kurihara and Sato (2006) applied variational inference to PCFGs. Their algorithm is similar to ours, but they did not consider nonparametric models.

5 Conclusion

We have presented the HDP-PCFG, a nonparametric Bayesian model for PCFGs, along with an efficient variational inference algorithm. While our primary contribution is the elucidation of the model and algorithm, we have also explored some important empirical properties of the HDP-PCFG and also demonstrated the potential of variational HDP-PCFGs on a full-scale parsing task.

References

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2:1152–1174.

M. Beal, Z. Ghahramani, and C. Rasmussen. 2002. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–584.

D. Bertsekas. 1999. *Nonlinear programming*.

D. Blei and M. I. Jordan. 2005. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1:121–144.

E. Charniak. 1996. Tree-bank grammars. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *North American Association for Computational Linguistics (NAACL)*, pages 132–139.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

M. D. Escobar and M. West. 1995. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588.

T. S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230.

J. R. Finkel, T. Grenager, and C. Manning. 2007. The infinite tree. In *Association for Computational Linguistics (ACL)*.

S. Goldwater, T. Griffiths, and M. Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

H. Ishwaran and L. F. James. 2001. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96:161–173.

M. Johnson, T. Griffiths, and S. Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems (NIPS)*.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Association for Computational Linguistics (ACL)*, pages 423–430.

K. Kurihara and T. Sato. 2004. An application of the variational Bayesian approach to probabilistic context-free grammars. In *International Joint Conference on Natural Language Processing Workshop Beyond Shallow Analyses*.

K. Kurihara and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*.

P. Liang, S. Petrov, M. I. Jordan, and D. Klein. 2007. Nonparametric PCFGs using Dirichlet processes. Technical report, Department of Statistics, University of California at Berkeley.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Association for Computational Linguistics (ACL)*.

S. Petrov and D. Klein. 2007. Learning and inference for hierarchically split PCFGs. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

J. Pitman. 2002. Combinatorial stochastic processes. Technical Report 621, Department of Statistics, University of California at Berkeley.

J. Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

A. Stolcke and S. Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In *Grammatical Inference and Applications*.

Y. W. Teh, M. I. Jordan, M. Beal, and D. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

Exploiting Wikipedia as External Knowledge for Named Entity Recognition

Jun'ichi Kazama and Kentaro Torisawa
Japan Advanced Institute of Science and Technology (JAIST)
Asahidai 1-1, Nomi, Ishikawa, 923-1292 Japan
{kazama, torisawa}@jaist.ac.jp

Abstract

We explore the use of Wikipedia as external knowledge to improve named entity recognition (NER). Our method retrieves the corresponding Wikipedia entry for each candidate word sequence and extracts a category label from the first sentence of the entry, which can be thought of as a definition part. These category labels are used as features in a CRF-based NE tagger. We demonstrate using the CoNLL 2003 dataset that the Wikipedia category labels extracted by such a simple method actually improve the accuracy of NER.

1 Introduction

It has been known that *Gazetteers*, or entity dictionaries, are important for improving the performance of named entity recognition. However, building and maintaining high-quality gazetteers is very time consuming. Many methods have been proposed for solving this problem by automatically extracting gazetteers from large amounts of texts (Riloff and Jones, 1999; Thelen and Riloff, 2002; Etzioni et al., 2005; Shinzato et al., 2006; Talukdar et al., 2006; Nadeau et al., 2006). However, these methods require complicated induction of patterns or statistical methods to extract high-quality gazetteers.

We have recently seen a rapid and successful growth of Wikipedia (http://www.wikipedia.org), which is an open, collaborative encyclopedia on the Web. Wikipedia has now more than 1,700,000 articles on the English version (March 2007) and the number is still increasing. Since Wikipedia aims to be an encyclopedia, most articles are about named entities and they are more structured than raw texts. Although it cannot be used as gazetteers directly since it is not intended as a machine readable resource, extracting knowledge such as gazetteers from Wikipedia will be much easier than from raw texts or from usual Web texts because of its structure. It is also important that Wikipedia is updated every day and therefore new named entities are added constantly. We think that extracting knowledge from Wikipedia for natural language processing is one of the promising ways towards enabling large-scale, real-life applications. In fact, many studies that try to exploit Wikipedia as a knowledge source have recently emerged (Bunescu and Paşca, 2006; Toral and Muñoz, 2006; Ruiz-Casado et al., 2006; Ponzetto and Strube, 2006; Strube and Ponzetto, 2006; Zesch et al., 2007).

As a first step towards such approach, we demonstrate in this paper that category labels extracted from the first sentence of a Wikipedia article, which can be thought of as the *definition* of the entity described in the article, are really useful to improve the accuracy of NER. For example, "Franz Fischler" has the article with the first sentence, "Franz Fischler (born September 23, 1946) is an Austrian politician." We extract "politician" from this sentence as the category label for "Franz Fischler". We use such category labels as well as matching information as features of a CRF-based NE tagger. In our experiments using the CoNLL 2003 NER dataset (Tjong et al., 2003), we demonstrate that we can improve performance by using the Wikipedia features by 1.58 points in F-measure from the baseline, and by 1.21 points from the model that only uses the gazetteers provided in the CoNLL 2003 dataset. Our final model incorporating all features achieved 88.02 in F-measure, which means a 3.03 point improvement over the baseline, which does not use any

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 698–707, Prague, June 2007.

gazetteer-type feature.

The studies most relevant to ours are Bunescu and Paşca (2006) and Toral and Muñoz (2006).

Bunescu and Paşca (2006) presented a method of disambiguating ambiguous entities exploiting internal links in Wikipedia as training examples. The difference however is that our method tries to use Wikipedia features for NER, not for disambiguation which assumes that entity regions are already found. They also did not focus on the first sentence of an article. Also, our method does not disambiguate ambiguous entities, since accurate disambiguation is difficult and possibly introduces noise. There are two popular ways for presenting ambiguous entities in Wikipedia. The first is to redirect users to a disambiguation page, and the second is to redirect users to one of the articles. We only focused on the second case and did not utilize disambiguation pages in this study. This method is simple but works well because the article presented in the second case represents in many cases the major meaning of the ambiguous entities and therefore that meaning frequently appears in a corpus.

Toral and Muñoz (2006) tried to extract gazetteers from Wikipedia by focusing on the first sentences. However, their way of using the first sentence is slightly different. We focus on the first noun phrase after *be* in the first sentence, while they used all the nouns in the sentence. By using these nouns and WordNet, they tried to map Wikipedia entities to abstract categories (e.g., LOC, PER ORG, MISC) used in usual NER datasets. We on the other hand use the obtained category labels directly as features, since we think the mapping performed automatically by a CRF model is more precise than the mapping by heuristic methods. Finally, they did not demonstrate the usefulness of the extracted gazetteers in actual NER systems.

The rest of the paper is organized as follows. We first explain the structure of Wikipedia in Section 2. Next, we introduce our method of extracting and using category labels in Section 3. We then show the experimental results on the CoNLL 2003 NER dataset in Section 4. Finally, we discuss the possibility of further improvement and future work in Section 5.

2 Wikipedia

2.1 Basic structure

An article in Wikipedia is identified by a unique name, which can be obtained by concatenating the words in the article title with underscore "_". For example, the unique name for the article, "David Beckham", is David_Beckham. We call these unique names "entity names" in this paper.

Wikipedia articles have many useful structures for knowledge extraction such as headings, lists, internal links, categories, and tables. These are marked up by using the Wikipedia syntax in source files, which authors edit. See the Wikipedia entry identified by How_to_edit_a_page for the details of the markup language.

We describe two important structures, redirections and disabiguation pages, in the following sections.

2.2 Redirection

Some entity names in Wikipedia do not have a substantive article and are only redirected to an article with another entity name. This mechanism is called "redirection". Redirections are marked up as "#REDIRECT [[A B C]]" in source files, where "[[...]]" is a syntax for a link to another article in Wikipedia (internal links). If the source file has such a description, users are automatically redirected to the article specified by the entity name in the brackes (A_B_C for the above example). Redirections are used for several purposes regarding ambiguity. For example, they are used for spelling resolution such as from "Apples" to "Apple" and abbreviation resolution such as from "MIT" to "Massachusetts Institute of Technology". They are also used in the context of more difficult disambiguations described in the next section.

2.3 Disambiguation pages

Some authors make a "disambiguation" page for an ambiguous entity name.[1] A disambiguation page typically enumerates possible articles for that name. For example, the page for "Beckham" enumerates "David Beckham (English footballer)", "Victoria

[1] We mean by "ambiguous" the case where a name can be used to refer to several difference entities (i.e., articles in Wikipedia).

Beckham (English celebrity and wife of David)", "Brice Beckham (American actor)", and so on. Most, but not all, disambiguation pages have a name like Beckham_(disambiguation) and are sometimes used with redirection. For example, Beckham is redirected to Beckham_(disambiguation) in the above example. However, it is also possible that Beckham redirects to one of the articles (e.g, David_Beckham). As we mentioned, we did not utilize the disambiguation pages and relied on the above case in this study.

2.4 Data

Snapshots of the entire contents of Wikipedia are provided in XML format for each language version. We used the English version at the point of February 2007, which includes 4,030,604 pages.[2] We imported the data into a text search engine[3] and used it for the research.

3 Method

In this section, we describe our method of extracting category labels from Wikipedia and how to use those labels in a CRF-based NER model.

3.1 Generating search candidates

Our purpose here is to find the corresponding entity in Wikipedia for each word sequence in a sentence. For example, given the sentence, "Rare Jimi Hendrix song draft sells for almost $17,000", we would like to know that "Jimi Hendrix" is described in Wikipedia and extract the category label, "musician", from the article. However, considering all possible word sequences is costly. We thus restricted the candidates to be searched to the word sequences of no more than eight words that start with a word containing at least one capitalized letter.[4]

3.2 Finding category labels

We converted a candidate word sequence to a Wikipedia entity name by concatenating the words with underscore. For example, a word sequence "Jimi Hendrix" is converted to Jimi_Hendrix. Next, we retrieved the article corresponding to the entity name.[5] If the page for the entity name is a redirection page, we followed redirection until we find a non-redirection page.

Although there is no strict formatting rule in Wikipedia, the convention is to start an article with a short sentence defining the entity the article describes. For example, the article for Jimi_Hendrix starts with the sentence, "Jimi Hendrix (November 27, 1942, Seattle, Washington - September 18, 1970, London, England) was an American guitarist, singer and songwriter." Most of the time, the head noun of the noun phrase just after *be* is a good category label. We thus tried to extract such head nouns from the articles.

First, we eliminated unnecessary markup such as italics, bold face, and internal links from the article. We also converted the markup for internal links like `[[Jimi Hendrix|Hendrix]]` to `Hendrix`, since the part after `|`, if it exists, represents the form to be displayed in the page. We also eliminated template markup, which is enclosed by `{{` and `}}`, because template markup sometimes comes at the beginning of the article and makes the extraction of the first sentence impossible.[6] We then divided the article into lines according to the new line code, `\n`, `<br>` HTML tags, and a very simple sentence segmentation rule for period (.). Next, we removed lines that match regular expression `/^\s*:/` to eliminate the lines such as:

> *This article is about the tree and its fruit.*
> *For the consumer electronics corporation,*
> *see Apple Inc.*

These sentences are not the content of the article but often placed at the beginning of an article. Fortunately, they are usually marked up using `:`, which is for indentation.

After the preprocessing described above, we extracted the first line in the remaining lines as the first sentence from which we extract a category label.

[2] The number of article pages is 2,954,255 including redirection pages

[3] We used HyperEstraier available at http://hyperestraier.sourceforge.net/index.html

[4] Words such as "It" and "He" are not considered as capitalized words here (we made a small list of stop words).

[5] There are pages for other than usual articles in the Wikipedia data. They are distinguished by a *namespace* attribute. To retrieve articles, we only searched in namespace 0, which is for usual articles.

[6] Templates are used for example to generate profile tables for persons.

We then performed POS tagging and phrase chunking. TagChunk (Daumé III and Marcu, 2005)[7] was used as a POS/chunk tagger. Next, we extracted the first noun phrase after the first "is", "was", "are", or "were" in the sentence. Basically, we extracted the last word in the noun phrase as the category label. However, we used the second noun phrase when the first noun phrase ended with "one", "kind", "sort", or "type", or it ended with "name" followed by "of". These rules were for treating examples like:

Jazz is [a kind]$_{\text{NP}}$ [of]$_{\text{PP}}$ [music]$_{\text{NP}}$ characterized by swung and blue notes.

In these cases, we would like to extract the head noun of the noun phrase after "of" (e.g., "music" in instead of "kind" for the above example). However, we would like to extract "name" itself when the sentence was like "Ichiro is a Japanese given name".

We did not utilize Wikipedia's "Category" sections in this study, since a Wikipedia article can have more than one category, and many of them are not clean hypernyms of the entity as far as we observed. We will need to select an appropriate category from the listed categories in order to utilize the Category section. We left this task for future research.

3.3 Using category labels as features

If we could find the category label for the candidate word sequence, we annotated it using IOB2 tags in the same way as we represent named entities. In IOB2 tagging, we use "B-X", "I-X", and "O" tags, where "B", "I", and "O" means the beginning of an entity, the inside of an entity, and the outside of entities respectively. Suffix X represents the category of an entity.[8] In this case, we used the extracted category label as the suffix. For example, if we found that "Jimi Hendrix" was in Wikipedia and extracted "guitarist" as the category label, we annotated the sentence, "Rare Jimi Hendrix song draft sells for almost $17,000", as:

Rare$_{\text{O}}$ Jimi$_{\text{B-guitarist}}$ Hendrix$_{\text{I-guitarist}}$ song$_{\text{O}}$ draft$_{\text{O}}$ for$_{\text{O}}$ almost$_{\text{O}}$ \$17,000$_{\text{O}}$.$_{\text{O}}$

Note that we adopted the leftmost longest match if there were several possible matchings. These IOB2 tags were used in the same way as other features in our NE tagger using Conditional Random Fields (CRFs) (Lafferty et al., 2001). For example, we used a feature such as "the Wikipedia tag is B-guitarist and the NE tag is B-PER".

4 Experiments

In this section, we demonstrate the usefulness of the extracted category labels for NER.

4.1 Data and setting

We used the English dataset of the CoNLL 2003 shared task (Tjong et al., 2003). It is a corpus of English newspaper articles, where four entity categories, PER, LOC, ORG, and MISC are annotated. It consists of training, development, and testing sets (14,987, 3,466, and 3,684 sentences, respectively). We concatenated the sentences in the same document according to the document boundary markers provided in the dataset.[9] This generated 964 documents for the training set, 216 documents for the development set, and 231 documents for the testing set. Although automatically assigned POS and chunk tags are also provided in the dataset, we used TagChunk (Daumé III and Marcu, 2005)[10] to assign POS and chunk tags, since we observed that accuracy could be improved, presumably due to the quality of the tags.[11]

We used the features summarized in Table 1 as the baseline feature set. These are similar to those used in other studies on NER. We omitted features whose surface part described in Table 1 occurred less than twice in the training corpus.

Gazetteer files for the four categories, PER (37,831 entries), LOC (10,069 entries), ORG (3,439 entries), and MISC (3,045 entries), are also provided in the dataset. We compiled these files into one gazetteer, where each entry has its entity category, and used it in the same way as the Wikipedia feature described in Section 3.3. We will compare features using this gazetteer with those using Wikipedia in the following experiments.

[7] http://www.cs.utah.edu/~hal/TagChunk/

[8] We use bare "B", "I", and "O" tags if we want to represent only the matching information.

[9] We used sentence concatenation because we found it improves the accuracy in another study (Kazama and Torisawa, 2007).

[10] http://www.cs.utah.edu/~hal/TagChunk/

[11] This is not because TagChunk overfits the CoNLL 2003 dataset (TagChunk is trained on the Penn Treebank (Wall Street Journal), while the CoNLL 2003 data are taken from the Reuters corpus).

Table 1: Baseline features. The value of a node feature is determined from the current label, y_0, and a surface feature determined only from $\boldsymbol{x}$. The value of an edge feature is determined by the previous label, y_{-1}, the current label, y_0, and a surface feature. Used surface features are the word (w), the downcased word (wl), the POS tag (pos), the chunk tag (chk), the prefix of the word of length n (pn), the suffix (sn), the word form features: 2d - cp (these are based on (Bikel et al., 1999))

Node features: $\{"", \mathrm{x}_{-2}, \mathrm{x}_{-1}, \mathrm{x}_0, \mathrm{x}_{+1}, \mathrm{x}_{+2}\} \times y_0$
x = w, wl, pos, chk, p1, p2, p3, p4, s1, s2, s3, s4, 2d, 4d, d&a, d&-, d&/, d&,, d&., n, ic, ac, l, cp
Edge features: $\{"", \mathrm{x}_{-2}, \mathrm{x}_{-1}, \mathrm{x}_0, \mathrm{x}_{+1}, \mathrm{x}_{+2}\} \times y_{-1} \times y_0$
x = w, wl, pos, chk, p1, p2, p3, p4, s1, s2, s3, s4, 2d, 4d, d&a, d&-, d&/, d&,, d&., n, ic, ac, l, cp
Bigram node features: $\{\mathrm{x}_{-2}\mathrm{x}_{-1}, \mathrm{x}_{-1}\mathrm{x}_0, \mathrm{x}_0\mathrm{x}_{+1}\} \times y_0$
x = wl, pos, chk
Bigram edge features: $\{\mathrm{x}_{-2}\mathrm{x}_{-1}, \mathrm{x}_{-1}\mathrm{x}_0, \mathrm{x}_0\mathrm{x}_{+1}\} \times y_{-1} \times y_0$
x = wl, pos, chk

We used CRF++ (ver. 0.44)[12] as the basis of our implementation of CRFs. We implemented scaling, which is similar to that for HMMs (see for instance (Rabiner, 1989)), in the forward-backward phase of CRF training to deal with long sequences due to sentence concatenation.[13] We used Gaussian regularization to avoid overfitting. The parameter of the Gaussian, σ^2, was tuned using the development set.[14] We stopped training when the relative change in the log-likelihood became less than a pre-defined threshold, 0.0001, for at least three iterations.

4.2 Category label finding

Table 2: Statistics of category label finding.

search candidates (including duplication)	256,418
candidates having Wikipedia article	39,258
(articles found by redirection)	9,587
first sentence found	38,949
category label extracted	23,885
(skipped "one")	544
(skipped "kind")	14
(skipped "sort")	1
(skipped "type")	41
(skipped "name of")	463
distinct category labels	1,248

Table 3: Examples of category labels (top 20).

category	**frequency**	**# distinct entities**
country	2598	152
city	1436	284
name	1270	281
player	578	250
day	564	131
month	554	15
club	537	167
surname	515	185
capital	454	79
state	416	60
term	369	78
form	344	40
town	287	97
cricketer	276	97
adjective	260	6
golfer	229	88
world	221	24
team	220	52
organization	214	38
second	212	1

Table 2 summarizes the statistics of category label finding for the training set. Table 3 lists examples of the extracted categories. As can be seen, we could extract more than 1,200 distinct category labels. These category labels seem to be useful, although there is no guarantee that the extracted category label is correct for each candidate.

4.3 Feature comparison

We compared the following features in this experiment.

Gazetteer Match (gaz_m) This feature represents the matching with a gazetteer entry by using "B", "I", and "O" tags. That is, this is the gazetteer version of **wp_m** below.

Gazetteer Category Label (gaz_c) This feature represents the matching with a gazetteer entry and its category by using "B-X", "I-X", and "O" tags, where X is one of "PER", "LOC", "ORG", and "MISC". That is, this is the gazetteer version of **wp_c** below.

Wikipedia Match (wp_m) This feature represents the matching with a Wikipedia entity by using "B", "I", and "O" tags.

[12] http://chasen.org/˜taku/software/CRF++

[13] We also replaced the optimization module in the original package with that used in the Amis maximum entropy estimator (http://www-tsujii.is.s.u-tokyo.ac.jp/amis) since we encountered problems with the provided module in some cases. Although this Amis module implements BLMVM (Benson and Moré, 2001), which supports the bounding of weights, we did not use this feature in this study (i.e., we just used it as the replacement for the L-BFGS optimizer in CRF++).

[14] We tested 15 points: $\{0.01, 0.02, 0.04, \ldots, 163.84, 327.68\}$.

Table 4: Statistics of gazetteer and Wikipedia features. Rows "NEs (%)" show the number of matches that also matched the regions of the named entities in the training data, and the percentage of such named entities (there were 23,499 named entities in total in the training data).

Gazetteer Match (gaz_m)	
matches	12,397
NEs (%)	6,415 (27.30%)
Wikipedia Match (wp_m)	
matches	27,779
NEs (%)	16,600 (70.64%)
Wikipedia Category Label (wp_c)	
matches	18,617
NEs (%)	11,645 (49.56%)
common with gazetteer match	5,664

Wikipedia Category Label (wp_c) This feature represents the matching with a Wikipedia entity and its category in the way described Section in 3.3. Note that this feature only fires when the category label is successfully extracted from the Wikipedia article.

For these **gaz_m**, **gaz_c**, **wp_m**, and **wp_c**, we generate the node features, the edge features, the bigram node features, and the bigram edge features, as described in Table 1.

Table 4 shows how many matches (the leftmost longest matches that were actually output) were found for **gaz_m**, **wp_m**, and **wp_c**. We omitted the numbers for **gaz_c**, since they are same as **gaz_m**. We can see that Wikipedia had more matches than the gazetteer, and covers more named entities (more than 70% of the NEs in the training corpus). The overlap between the gazetteer matches and the Wikipedia matches was moderate as the last row indicates (5,664 out of 18,617 matches). This indicates that Wikipedia has many entities that are not listed in the gazetteer.

We then compared the baseline model (**baseline**), which uses the feature set in Table 1, with the following models to see the effect of the gazetteer features and the Wikipedia features.

(A): + gaz_m This uses **gaz_m** in addition to the features in **baseline**.

(B): + gaz_m, gaz_c This uses **gaz_m** and **gaz_c** in addition to the features in **baseline**.

(C): + wp_m This uses **wp_m** in addition to the features in **baseline**.

(D): + wp_m, wp_c This uses **wp_m** and **wp_c** in addition to the features in **baseline**.

(E): + gaz_m, gaz_c, wp_m, wp_c This uses **gaz_m**, **gaz_c**, **wp_m**, and **wp_c** in addition to the features in **baseline**.

(F): + gaz_m, gaz_c, wp_m, wp_c (word comb.) This model uses the combination of words (wl) and **gaz_m**, **gaz_c**, **wp_m**, or **wp_c**, in addition to the features of model **(E)**. More specifically, these features are the node feature, $\mathrm{wl}_0 \times x_0 \times y_0$, the edge feature, $\mathrm{wl}_0 \times x_0 \times y_{-1} \times y_0$, the bigram node feature, $\mathrm{wl}_{-1} \times \mathrm{wl}_0 \times x_{-1} \times x_0 \times y_0$, and the bigram edge feature, $\mathrm{wl}_{-1} \times \mathrm{wl}_0 \times x_{-1} \times x_0 \times y_{-1} \times y_0$, where x is one of gaz_m, gaz_c, wp_m, and wp_c. We tested this model because we thought these combination features could alleviate the problem by incorrectly extracted categories in some cases, if there is a characteristic correlation between words and incorrectly extracted categories.

Table 5 shows the performance of these models. The results for (A) and (C) indicate that the matching information alone does not improve accuracy. This is because entity regions can be identified fairly correctly if models are trained using a sufficient amount of training data. The category labels, on the other hand, are actually important for improvement as the results for (B) and (D) indicate. The gazetteer model, (B), improved F-measure by 1.47 points from the baseline. The Wikipedia model, (D), improved F-measure by 1.58 points from the baseline. The effect of the gazetteer feature, **gaz_c**, and the Wikipedia features, **wp_c**, did not differ much. However, it is notable that the Wikipedia feature, which is obtained by our very simple method, achieved such an improvement easily.

The results for model (E) show that we can improve accuracy further, by using the gazetteer features and the Wikipedia features together. Model (E) achieved 87.67 in F-measure, which is better than those of (B) and (D). This result coincides with the fact that the overlap between the gazetteer feature

Table 5: Effect of gazetteer and Wikipedia features.

		dev			**eval**		
model (best σ^2)	category	**P**	**R**	**F**	**P**	**R**	**F**
baseline (20.48)	PER	90.29	92.89	91.57	87.19	91.34	89.22
	LOC	93.32	92.81	93.07	88.14	88.25	88.20
	ORG	85.36	83.07	84.20	82.25	78.93	80.55
	MISC	92.21	84.71	88.30	79.58	75.50	77.49
	ALL	90.42	89.38	**89.90**	85.17	84.81	**84.99**
(A): + gaz_m (81.92)	PER	90.60	92.56	91.57	87.90	90.72	89.29
	LOC	92.84	93.20	93.02	88.26	88.37	88.32
	ORG	85.54	82.92	84.21	82.37	79.05	80.68
	MISC	92.15	85.25	88.56	78.73	75.93	77.30
	ALL	90.41	89.45	**89.92**	85.33	84.76	**85.04**
(B): + gaz_m, gaz_c (163.84)	PER	92.45	94.41	93.42	90.78	91.96	91.37
	LOC	94.43	94.07	94.25	89.98	89.33	89.65
	ORG	86.68	85.38	86.03	82.43	81.34	81.88
	MISC	92.47	85.25	88.71	79.50	76.78	78.12
	ALL	91.77	90.84	**91.31**	86.74	86.17	**86.46**
(C): + wp_m (163.84)	PER	90.84	92.56	91.69	87.77	90.11	88.92
	LOC	92.63	93.03	92.83	87.23	88.07	87.65
	ORG	86.19	83.74	84.95	81.77	79.65	80.70
	MISC	91.69	84.92	88.18	79.04	75.21	77.08
	ALL	90.49	89.53	**90.01**	84.85	84.58	**84.71**
(D): + wp_m, wp_c (163.84)	PER	91.57	94.41	92.97	90.13	92.02	91.06
	LOC	94.78	93.96	94.37	89.41	89.63	89.52
	ORG	87.36	85.01	86.17	82.70	82.00	82.35
	MISC	91.87	84.60	88.09	81.34	76.35	78.77
	ALL	91.68	90.63	**91.15**	86.71	86.42	**86.57**
(E): + gaz_m, gaz_c, wp_m, wp_c (40.96)	PER	93.32	95.49	94.39	92.28	93.14	92.71
	LOC	94.91	94.39	94.65	90.69	90.47	90.58
	ORG	88.27	86.95	87.60	83.08	83.68	83.38
	MISC	93.14	85.36	89.08	81.33	76.92	79.06
	ALL	92.65	91.65	**92.15**	87.79	87.55	**87.67**
(F): + gaz_m, gaz_c, wp_m, wp_c (word comb.) (5.12)	PER	93.38	95.66	94.50	92.52	93.26	92.89
	LOC	94.88	94.77	94.83	91.25	90.71	90.98
	ORG	88.67	86.95	87.80	83.61	84.17	83.89
	MISC	93.56	85.03	89.09	81.63	77.21	79.36
	ALL	92.82	91.77	**92.29**	88.21	87.84	**88.02**

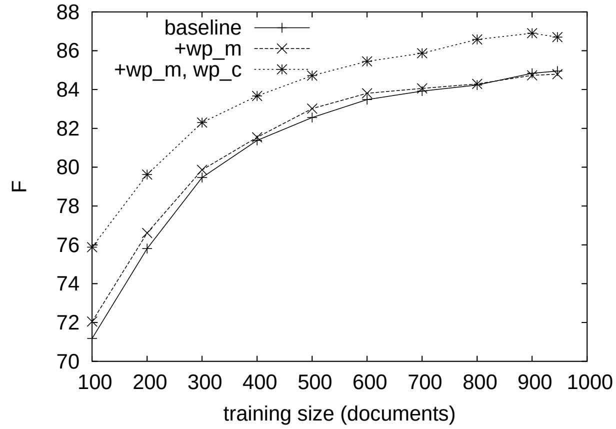


Figure 1: Relation between the training size and the accuracy.

and the Wikipedia feature was not so large. If we consider model (B) a practical baseline, we can say that the Wikipedia features improved the accuracy in F-measure by 1.21 points.

We can also see that the effect of the gazetteer features and the Wikipedia features were consistent irrespective of categories (i.e., PER, LOC, ORG, or MISC) and performance measures (i.e., precision, recall, or F-measure). This indicates that gazetteer-type features are reliable as features for NER.

The final model, (F), achieved 88.02 in F-measure. This is greater than that of the baseline by 3.03 points, showing the usefulness of the gazetteer type features.

4.4 Effect of training size

We observed in the previous experiment that the matching information alone was not useful. However, the situation may change if the size of the training data becomes small. We thus observed the effect of the training size for the Wikipedia features **wp_m** and **wp_c** (we used $\sigma^2 = 10.24$). Figure 1 shows the result. As can be seen, the matching information had a slight positive effect when the size of training data was small. For example, it improved F-measure by 0.8 points from the baseline at 200 documents. However, the superiority of category labels over the matching information did not change. The effect of category labels became greater as the training size became smaller. Its effect compared with the matching information alone was 3.01 points at 200 documents, while 1.91 points at 964 documents (i.e., the whole training data).

Table 6: Breakdown of improvements and errors.

(B) → (E)	num.	$\bar{g} \wedge \bar{w}$	$\bar{g} \wedge w$	$g \wedge \bar{w}$	$g \wedge w$
inc → **inc**	442	219	123	32	68
inc → **cor**	102	28	56	3	15
cor → **inc**	56	28	13	7	8
cor → **cor**	5,342	1,320	1,662	723	1,637

4.5 Improvement and error analysis

We analyze the improvements and the errors caused by using the Wikipedia features in this section.

We compared the output of (B) and (E) for the development set. There were 5,942 named entities in the development set. We assessed how the labeling for these entities changed between (B) and (E). Note that the labeling for 199 sentences out of total 3,466 sentences was changed. Table 6 shows the breakdown of the improvements and the errors. "**inc**" in the table means that the model could not label the entity correctly, i.e., the model could not find the entity region at all, or it assigned an incorrect category to the entity. "**cor**" means that the model could label the entity correctly. The column, "**inc** → **cor**", for example, has the numbers for the entities that were labeled incorrectly by (B) but labeled correctly by (E). We can see from the column, "num", that the number of improvements by (E) exceeded the number of errors introduced by (E) (102 vs. 56). Table 6 also shows how the gazetteer feature, **gaz_c**, and the Wikipedia feature, **wp_c**, fired in each case. We mean that the gazetteer feature fired by using "g", and that the Wikipedia feature fired by using "w". "$\bar{g}$" and "$\bar{w}$" mean that the feature did not fire. As is the case for other machine learning methods, it is difficult to find a clear reason for each improvement or error. However, we can see that the number of $\bar{g} \wedge w$ exceeded those of other cases in the case of "**inc** → **cor**", meaning that the Wikipedia feature contributed the most.

Finally, we show an example of case **inc** → **cor** in Figure 2. We can see that "Gazzetta dello Sport" in the sentence was correctly labeled as an entity of "ORG" category by model (E), because the Wikipedia feature identified it as a newspaper entity.[15]

[15] Note that the category label, "character", for "Atalanta" in the sentence was not correct in this context, which is an example where disambiguation is required. The final recognition was correct in this case presumably because of the information from **gaz_c** feature.

The	Gazzetta	dello	Sport	said	the	deal	would	cost	Atalanta	around	$	600,000	.	
O	O	O	B-ORG	O	O	O	O	O	B-ORG	O	O	O	O	**- gaz_c**
O	B-newspaper	I-newspaper	I-newspaper	O	O	O	O	O	B-character	O	O	O	O	**- wp_c**
O	B-ORG	I-ORG	I-ORG	O	O	O	O	O	B-ORG	O	O	O	O	**- correct**
O	B-LOC	O	B-ORG	O	O	O	O	O	B-ORG	O	O	O	O	**- (B)**
O	B-ORG	I-ORG	I-ORG	O	O	O	O	O	B-ORG	O	O	O	O	**- (E)**

Figure 2: An example of improvement caused by Wikipedia feature.

5 Discussion and Future Work

We have empirically shown that even category labels extracted from Wikipedia by a simple method such as ours really improves the accuracy of a NER model. The results indicate that structures in Wikipedia are suited for knowledge extraction. However, the results also indicate that there is room for improvement, considering that the effects of **gaz_c** and **wp_c** were similar, while the matching rate was greater for **wp_c**. An issue, which we should treat, is the disambiguation of ambiguous entities. Our method worked well although it was very simple, presumably because of the following reason. (1) If a retrieved page is a disambiguation page, we cannot extract a category label and critical noise is not introduced. (2) If a retrieved page is not a disambiguation page, it will be the page describing the major meaning determined by the agreement of many authors. The extracted categories are useful for improving accuracy because the major meaning will be used frequently in the corpus. However, it is clear that disambiguation techniques are required to achieve further improvements. In addition, if Wikipedia grows at the current rate, it is possible that almost all entities become ambiguous and a retrieved page is a disambiguation page most of the time. We will need a method for finding the most suitable article from the articles listed in a disambiguation page.

An interesting point in our results is that Wikipedia category labels improved accuracy, although they were much more specific (more than 1,200 categories) than the four categories of the CoNLL 2003 dataset. The correlation between a Wikipedia category label and a category label of NER (e.g., "musician" to "PER") was probably learned by a CRF tagger. However, the merit of using such specific Wikipedia labels will be much greater when we aim at developing NER systems for more fine-grained NE categories such as proposed in Sekine et al. (2002) or Shinzato et al. (2006). We thus would like to investigate the effect of the Wikipedia feature for NER with such fine-grained categories as well. Disambiguation techniques will be important again in that case. Although the impact of ambiguity will be small as long as the target categories are abstract and an incorrectly extracted category is in the same abstract category as the correct one (e.g., extracting "footballer" instead of "cricketer"), such mis-categorization is critical if it is necessary to distinguish footballers from cricketers.

6 Conclusion

We tried to exploit Wikipedia as external knowledge to improve NER. We extracted a category label from the first sentence of a Wikipedia article and used it as a feature of a CRF-based NE tagger. The experiments using the CoNLL 2003 NER dataset demonstrated that category labels extracted by such a simple method really improved accuracy. However, disambiguation techniques will become more important as Wikipedia grows or if we aim at more fine-grained NER. We thus would like to incorporate a disambiguation technique into our method in future work. Exploiting Wikipedia structures such as disambiguation pages and link structures will be the key in that case as well.

References

S. J. Benson and J. J. Moré. 2001. A limited memory variable metric method for bound constraint minimization. Technical Report ANL/MCS-P909-0901, Argonne National Laboratory.

D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

R. Bunescu and M. Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL 2006*.

H. Daumé III and D. Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML 2005*.

O. Etzioni, M. Cafarella, D. Downey, A. M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web – an experimental study. *Artificial Intelligence Journal*.

J. Kazama and K. Torisawa. 2007. A new perceptron algorithm for sequence labeling with non-local features. In *EMNLP-CoNLL 2007*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, pages 282–289.

D. Nadeau, Peter D. Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *19th Canadian Conference on Artificial Intelligence*.

S. P. Ponzetto and M. Strube. 2006. Exploiting semantic role lebeling, WordNet and Wikipedia for coreference resolution. In *NAACL 2006*.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *16th National Conference on Artificial Intelligence (AAAI-99)*.

M. Ruiz-Casado, E. Alfonseca, and P. Castells. 2006. From Wikipedia to semantic relationships: a semi-automated annotation approach. In *Third European Semantic Web Conference (ESWC 2006)*.

S. Sekine, K. Sudo, and C. Nobata. 2002. Extended named entity hierarchy. In *LREC '02*.

K. Shinzato, S. Sekine, N. Yoshinaga, and K. Torisawa. 2006. Constructing dictionaries for named entity recognition on specific domains from the Web. In *Web Content Mining with Human Language Technologies Workshop on the 5th International Semantic Web*.

M. Strube and S. P. Ponzetto. 2006. WikiRelate! computing semantic relatedness using Wikipedia. In *AAAI 2006*.

P. P. Talukdar, T. Brants, M. Liberman, and F. Pereira. 2006. A context pattern induction method for named entity extraction. In *CoNLL 2006*.

M. Thelen and E. Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern context. In *EMNLP 2002*.

E. F. Tjong, K. Sang, and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL 2003*.

A. Toral and R. Muñoz. 2006. A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia. In *EACL 2006*.

T. Zesch, I. Gurevych, and M. Möhlhäuser. 2007. Analyzing and accessing Wikipedia as a lexical semantic resource. In *Biannual Conference of the Society for Computational Linguistics and Language Technology*.

Large-Scale Named Entity Disambiguation Based on Wikipedia Data

Silviu Cucerzan
Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
silviu@microsoft.com

Abstract

This paper presents a large-scale system for the recognition and semantic disambiguation of named entities based on information extracted from a large encyclopedic collection and Web search results. It describes in detail the disambiguation paradigm employed and the information extraction process from Wikipedia. Through a process of maximizing the agreement between the contextual information extracted from Wikipedia and the context of a document, as well as the agreement among the category tags associated with the candidate entities, the implemented system shows high disambiguation accuracy on both news stories and Wikipedia articles.

1 Introduction and Related Work

The ability to identify the *named entities* (such as people and locations) has been established as an important task in several areas, including topic detection and tracking, machine translation, and information retrieval. Its goal is the identification of mentions of entities in text (also referred to as *surface forms* henceforth), and their labeling with one of several entity type labels. Note that an entity (such as George W. Bush, the current president of the U.S.) can be referred to by multiple surface forms (e.g., “George Bush” and “Bush”) and a surface form (e.g., “Bush”) can refer to multiple entities (e.g., two U.S. presidents, the football player Reggie Bush, and the rock band called Bush).

When it was introduced, in the 6th Message Understanding Conference (Grishman and Sundheim, 1996), the named entity recognition task comprised three entity identification and labeling subtasks: ENAMEX (proper names and acronyms designating persons, locations, and organizations), TIMEX (absolute temporal terms) and NUMEX (numeric expressions, monetary expressions, and percentages). Since 1995, other similar named entity recognition tasks have been defined, among which CoNLL (e.g., Tjong Kim Sang and De Meulder, 2003) and ACE (Doddington et al., 2004). In addition to structural disambiguation (e.g., does “the Alliance for Democracy in Mali” mention one, two, or three entities?) and entity labeling (e.g., does “Washington went ahead” mention a person, a place, or an organization?), MUC and ACE also included a within document coreference task, of grouping all the mentions of an entity in a document together (Hirschman and Chinchor, 1997).

When breaking the document boundary and scaling entity tracking to a large document collection or the Web, resolving semantic ambiguity becomes of central importance, as many surface forms turn out to be ambiguous. For example, the surface form “Texas” is used to refer to more than twenty different named entities in Wikipedia. In the context “former Texas quarterback James Street”, Texas refers to the University of Texas at Austin; in the context “in 2000, Texas released a greatest hits album”, Texas refers to the British pop band; in the context “Texas borders Oklahoma on the north”, it refers to the U.S. state; while in the context “the characters in Texas include both real and fictional explorers”, the same surface form refers to the novel written by James A. Michener.

Bagga and Baldwin (1998) tackled the problem of cross-document coreference by comparing, for any pair of entities in two documents, the word vectors built from all the sentences containing mentions of the targeted entities. Ravin and Kazi (1999) further refined the method of solving coreference through measuring context similarity and integrated it into *Nominator* (Wacholder et al., 1997), which was one of the first successful systems for named entity recognition and co-reference resolution. However, both studies targeted the clustering of all mentions of an entity across a given document collection rather than the mapping of these mentions to a given reference list of entities.

A body of work that did employ reference entity lists targeted the resolution of geographic names in

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 708–716, Prague, June 2007. ©2007 Association for Computational Linguistics

text. Woodruff and Plaunt (1994) used a list of 80k geographic entities and achieved a disambiguation precision of 75%. Kanada (1999) employed a list of 96k entities and reported 96% precision for geographic name disambiguation in Japanese text. Smith and Crane (2002) used the Cruchley's and the Getty thesauri, in conjunction with heuristics inspired from the Nominator work, and obtained between 74% and 93% precision at recall levels of 89-99% on five different history text corpora. Overell and Rüger (2006) also employed the Getty thesaurus as reference and used Wikipedia to develop a co-occurrence model and to test their system.

In many respects, the problem of resolving ambiguous surface forms based on a reference list of entities is similar to the lexical sample task in word sense disambiguation (WSD). This task, which has supported large-scale evaluations – SENSEVAL 1-3 (Kilgarriff and Rosenzweig, 2000; Edmonds and Cotton, 2001; Mihalcea et al., 2004) – aims to assign dictionary meanings to all the instances of a predetermined set of polysemous words in a corpus (for example, choose whether the word "church" refers to a building or an institution in a given context). However, these evaluations did not include proper noun disambiguation and omitted named entity meanings from the targeted semantic labels and the development and test contexts (e.g., "Church and Gale showed that the frequency [..]").

The problem of resolving ambiguous names also arises naturally in Web search. For queries such as "Jim Clark" or "Michael Jordan", search engines return blended sets of results referring to many different people. Mann and Yarowsky (2003) addressed the task of clustering the Web search results for a set of ambiguous personal names by employing a rich feature space of biographic facts obtained via bootstrapped extraction patterns. They reported 88% precision and 73% recall in a three-way classification (most common, secondary, and other uses).

Raghavan et al. (2004) explored the use of entity language models for tasks such as clustering entities by profession and classifying politicians as liberal or conservative. To build the models, they recognized the named entities in the TREC-8 corpus and computed the probability distributions over words occurring within a certain distance of any instance labeled as Person of the canonical surface form of 162 famous people.

Our aim has been to build a named entity recognition and disambiguation system that employs a comprehensive list of entities and a vast amount of world knowledge. Thus, we turned our attention to the Wikipedia collection, the largest organized knowledge repository on the Web (Remy, 2002).

Wikipedia was successfully employed previously by Strube and Ponzetto (2006) and Gabrilovich and Markovitch (2007) to devise methods for computing semantic relatedness of documents, WikiRelate! and Explicit Semantic Analysis (ESA), respectively. For any pair of words, WikiRelate! attempts to find a pair of articles with titles that contain those words and then computes their relatedness from the word-based similarity of the articles and the distance between the articles' categories in the Wikipedia category tree. ESA works by first building an inverted index from words to all Wikipedia articles that contain them. Then, it estimates a relatedness score for any two documents by using the inverted index to build a vector over Wikipedia articles for each document and by computing the cosine similarity between the two vectors.

The most similar work to date was published by Bunescu and Paşca (2006). They employed several of the disambiguation resources discussed in this paper (Wikipedia entity pages, redirection pages, categories, and hyperlinks) and built a context-article cosine similarity model and an SVM based on a taxonomy kernel. They evaluated their models for person name disambiguation over 110, 540, and 2,847 categories, reporting accuracies between 55.4% and 84.8% on (55-word context, entity) pairs extracted from Wikipedia, depending on the model and the development/test data employed.

The system discussed in this paper performs both named entity identification and disambiguation. The entity identification and in-document coreference components resemble the Nominator system (Wacholder et al., 1997). However, while Nominator made heavy use of heuristics and lexical clues to solve the structural ambiguity of entity mentions, we employ statistics extracted from Wikipedia and Web search results. The disambiguation component, which constitutes the main focus of the paper, employs a vast amount of contextual and category information automatically extracted from Wikipedia over a space of 1.4 million distinct entities/concepts, making extensive use of the highly interlinked structure of this collection. We augment the Wikipedia category information with information automatically extracted from Wikipedia list pages and use it in conjunction with the context information in a vectorial model that employs a novel disambiguation method.

2 The Disambiguation Paradigm

We present in this section an overview of the proposed disambiguation model and the world knowledge data employed in the instantiation of the model discussed in this paper. The formal model is discussed in detailed in Section 5.

The world knowledge used includes the known entities (most articles in Wikipedia are associated to an entity/concept), their entity class when available (Person, Location, Organization, and Miscellaneous), their known surface forms (terms that are used to mention the entities in text), contextual evidence (words or other entities that describe or co-occur with an entity), and category tags (which describe topics to which an entity belongs to).

For example, Figure 1 shows nine of the over 70 different entities that are referred to as "Columbia" in Wikipedia and some of the category and contextual information associated with one of these entities, the Space Shuttle Columbia.

The disambiguation process uses the data associated with the known surface forms identified in a document and all their possible entity disambiguations to *maximize* the agreement between the context data stored for the candidate entities and the contextual information in the document, and also, the agreement among the category tags of the candidate entities. For example, a document that contains the surface forms "Columbia" and "Discovery" is likely to refer to the Space Shuttle Columbia and the Space Shuttle Discovery because these candidate entities share the category tags *LIST_astronomical_topics*, C*AT_Manned_spacecraft*, *CAT_Space_Shuttles* (the extraction of such tags is presented in Section 3.2), while other entity disambiguations, such as Columbia Pictures and Space Shuttle Discovery, do not share any common category tags. The agreement maximization process is discussed in depth in Section 5.

This process is based on the assumption that typically, all instances of a surface form in a document have the same meaning. Nonetheless, there are a non-negligible number of cases in which the *one sense per discourse* assumption (Gale et al., 1992) does not hold. To address this problem, we employ an iterative approach, of shrinking the context size used to disambiguate surface forms for which there is no *dominating* entity disambiguation at document level, performing the disambiguation at the paragraph level and then at the sentence level if necessary.

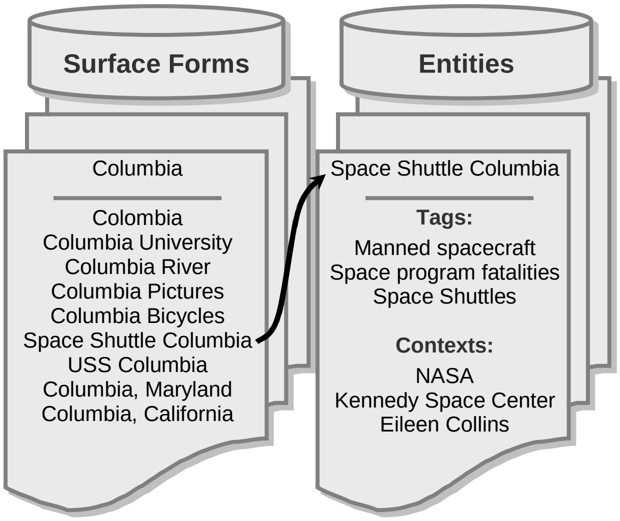


Figure 1. The model of storing the information extracted from Wikipedia into two databases.

3 Information Extraction from Wikipedia

We discuss now the extraction of entities and the three main types of disambiguation clues (entity surface forms, category tags, and contexts) used by the implemented system. While this information extraction was performed on the English version of the Wikipedia collection, versions in other languages or other collections, such as Encarta or WebMD, could be targeted in a similar manner.

When processing the Wikipedia collection, we distinguish among four types of articles: *entity pages*, *redirecting pages*, *disambiguation pages*, and *list pages*. The characteristics of these articles and the processing applied to each type to extract the three sets of clues employed by the disambiguation model are discussed in the next three subsections.

3.1 Surface Form to Entity Mappings

There are four sources that we use to extract entity surface forms: the titles of entity pages, the titles of redirecting pages, the disambiguation pages, and the references to entity pages in other Wikipedia articles. An *entity page* is an article that contains information focused on one single entity, such as a person, a place, or a work of art. For example, Wikipedia contains a page titled "Texas (TV series)", which offers information about the soap opera that aired on NBC from 1980 until 1982. A *redirecting page* typically contains only a reference to an entity page. For example, the article titled "Another World in Texas" contains a redirection to the article titled "Texas (TV series)". From these two articles, we extract the entity Texas (TV series) and its surface forms *Texas (TV series)*, *Texas* and *Another World in Texas*. As shown in this example, we store not only the exact article titles but also the corresponding forms from which we eliminate appositives (either within parentheses or following a comma).

We also extract surface form to entity mappings from Wikipedia disambiguation pages, which are specially marked articles having as title a surface form, typically followed by the word "disambiguation" (e.g., "Texas (disambiguation)"), and containing a list of references to pages for entities that are typically mentioned using that surface form.

Additionally, we extract all the surface forms used at least in two articles to refer to a Wikipedia entity page. Illustratively, the article for Pam Long contains the following *Wikitext*, which uses the surface form "Texas" to refer to Texas (TV series):

> After graduation, she went to [[New York City]] and played Ashley Linden on [[Texas (TV series)|Texas]] from [[1981]] to [[1982]].

In Wikitext, the references to other Wikipedia articles are within pairs of double square brackets. If a reference contains a vertical bar then the text at the left of the bar is the name of the referred article (e.g. "Texas (TV Series)"), while the text at the right of the bar (e.g., "Texas") is the surface form that is displayed (also referred to as the anchor text of the link). Otherwise, the surface form shown in the text is identical to the title of the Wikipedia article referred (e.g., "New York City").

Using these four sources, we extracted more than 1.4 million entities, with an average of 2.4 surface forms per entity. We obtained 377k entities with one surface form, 166k entities with two surface forms, and 79k entities with three surface forms. At the other extreme, we extracted one entity with no less than 99 surface forms.

3.2 Category Information

All articles that are titled "List of […]" or "Table of […]" are treated separately as *list pages*. They were built by Wikipedia contributors to group entities of the same type together (e.g., "List of anthropologists", "List of animated television series", etc.) and are used by our system to extract category tags for the entities listed in these articles. The tags are named after the title of the Wikipedia list page. For example, from the article "List of band name etymologies", the system extracts the category tag *LIST_band_name_etymologies* and labels all the entities referenced in the list, including Texas (band), with this tag. This process resulted in the extraction of more than 1 million (entity, tag) pairs. After a post-processing phase that discards temporal tags, as well as several types of non-useful tags such as "people by name" and "places by name", we obtained a filtered list of 540 thousand pairs.

We also exploit the fact that Wikipedia enables contributors to assign *categories* to each article, which are defined as "major topics that are likely to be useful to someone reading the article". Because any Wikipedia contributor can add a category to any article and the work of filtering out bogus assignments is tedious, these categories seem to be noisier than the lists, but they can still provide a tremendous amount of information. We extracted the categories of each entity page and assigned them as tags to the corresponding entity. Again, we employed some basic filtering to discard meta-categories (e.g., "Articles with unsourced statements") and categories not useful for the process of disambiguation through tag agreement (e.g., "Living people", "1929 births"). This extraction process resulted in 2.65 million (entity, tag) pairs over a space of 139,029 category tags.

We also attempted to extract category tags based on lexicosyntactic patterns, more specifically from enumerations of entities. For example, the paragraph titled "Music of Scotland" (shown below in Wikitext) in the Wikipedia article on Scotland contains an enumeration of entities, which can be labeled *ENUM_Scotland_PAR_Music_of_Scotland*:

> Modern Scottish [[pop music]] has produced many international bands including the [[Bay City Rollers]], [[Primal Scream]], [[Simple Minds]], [[The Proclaimers]], [[Deacon Blue]], [[Texas (band)|Texas]], [[Franz Ferdinand]], [[Belle and Sebastian]], and [[Travis (band)|Travis]], as well as individual artists such as [[Gerry Rafferty]], [[Lulu]], [[Annie Lennox]] and [[Lloyd Cole]], and world-famous Gaelic groups such as [[Runrig]] and [[Capercaillie (band)|Capercaillie]].

Lexicosyntactic patterns have been employed successfully in the past (e.g., Hearst, 1992; Roark and Charniak, 1998; Cederberg and Widdows, 2003), and this type of tag extraction is still a promising direction for the future. However, the brute force approach we tried – of indiscriminately tagging the entities of enumerations of four or more entities – was found to introduce a large amount of noise into the system in our development experiments.

3.3 Contexts

To extract contextual clues for an entity, we use the information present in that entity's page and in the other articles that explicitly refer to that entity.

First, the appositives in the titles of entity pages, which are eliminated to derive entity surface forms (as discussed in Section 3.1) are saved as contextual clues. For example, "TV series" becomes a context for the entity Texas (TV series).

We then extract all the entity references in the entity page. For example, from the article on Texas (band), for which a snippet in Wikitext is shown below, we extract as contexts the references pop music, Glasgow, Scotland, and so on:

> '''Texas''' is a [[pop music]] band from [[Glasgow]], [[Scotland]], [[United Kingdom]]. They were founded by [[Johnny McElhone]] in [[1986 in music|1986]] and had their performing debut in [[March]] [[1988]] at […]

Reciprocally, we also extract from the same article that the entity Texas (band) is a good context for pop music, Glasgow, Scotland, etc.

The number of contexts extracted in this manner is overwhelming and had to be reduced to a manageable size. In our development experiments, we explored various ways of reducing the context information, for example, by extracting only entities with a certain number of mentions in an article, or by discarding mentions with low TF*IDF scores (Salton, 1989). In the end, we chose a strategy in which we employ as contexts for an entity two category of references: those mentioned in the first paragraph of the targeted entity page, and those for which the corresponding pages refer back to the targeted entity. For example, Pam Long and Texas (TV series) are extracted as relevant contexts for each other because their corresponding Wikipedia articles reference one another – a relevant snippet from the Pam Long article is cited in Section 3.1 and a snippet from the article for Texas (TV series) that references Pam Long is shown below:

> In 1982 [[Gail Kobe]] became executive producer and [[Pam Long]] became headwriter.

In this manner, we extracted approximately 38 million (entity, context) pairs.

4 Document Analysis

In this section, we describe concisely the main text processing and entity identification components of the implemented system. We will then focus on the novel entity disambiguation component, which we propose and evaluate in this paper, in Section 5.

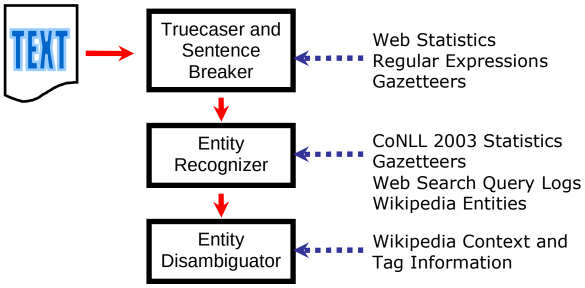


Figure 2. An overview of the processes employed by the proposed system.

Figure 2 outlines the processes and the resources that are employed by the implemented system in the analysis of text documents. First, the system splits a document into sentences and truecases the beginning of each sentence, hypothesizing whether the first word is part of an entity or it is capitalized because of orthographic conventions. It also identifies titles and hypothesizes the correct case for all words in the titles. This is done based on statistics extracted from a one-billion-word corpus, with back-off to Web statistics.

In a second stage, a hybrid named-entity recognizer based on capitalization rules, Web statistics, and statistics extracted from the CoNLL 2003 shared task data (Tjong Kim Sang and De Meulder, 2003) identifies the boundaries of the entity mentions in the text and assigns each set of mentions sharing the same surface form a probability distribution over four labels: Person, Location, Organization, and Miscellaneous.[1] The named entity recognition component resolves the structural ambiguity with regard to conjunctions (e.g., "Barnes and Noble", "Lewis and Clark"), possessives (e.g., "Alice's Adventures in Wonderland", "Britain's Tony Blair"), and prepositional attachment (e.g., "Whitney Museum of American Art", "Whitney Museum in New York") by using the surface form information extracted from Wikipedia, when available, with back-off to co-occurrence counts on the Web, in a similar way to Lapata and Keller (2004). Recursively, for each ambiguous term T_0 of the form T_1 *Particle* T_2, where *Particle* is one of a possessive pronoun, a coordinative conjunction, and a preposition, optionally followed by a determiner, and the terms T_1 and T_2 are se-

[1] While the named entity labels are used only to solve in-document coreferences by the current system, as described further in this section, preliminary experiments of probabilistically labeling the Wikipedia pages show that the these labels could also be used successfully in the disambiguation process.

quences of capitalized words and particles, we send to a search engine the query "″T_1″ ″T_2″", which forces the engine to return only documents in which the whole terms T_1 and T_2 appear. We then count the number of times the snippets of the top $N = 200$ search results returned contain the term T_0 and compare it with an empirically obtained threshold to hypothesize whether T_0 is the mention of one entity or encompasses the mentions of two entities, T_1 and T_2.

As Wacholder et al. (1997) noted, it is fairly common for one of the mentions of an entity in a document to be a long, typical surface form of that entity (e.g., "George W. Bush"), while the other mentions are shorter surface forms (e.g., "Bush"). Therefore, before attempting to solve the semantic ambiguity, the system hypothesizes in-document coreferences and maps short surface forms to longer surface forms with the same dominant label (for example, "Brown"/$_{\text{PERSON}}$ can be mapped to "Michael Brown"/$_{\text{PERSON}}$). Acronyms are also resolved in a similar manner when possible.

In the third stage, the contextual and category information extracted from Wikipedia is used to disambiguate the entities in the text. This stage is discussed formally in Section 5 and evaluated in Section 6. Note that the performance of the disambiguation component is meaningful only when most named entity mentions are accurately identified in text. Thus, we first measured the performance of the named entity recognition component on the CoNLL 2003 test set and obtained a competitive F-measure of 0.835 (82.2% precision and 84.8% recall).

Finally, the implemented system creates hyperlinks to the appropriate pages in Wikipedia. Figure 3 shows the output of the implemented system on a sample news story, in which the identified and disambiguated surface forms are hyperlinked to Wikipedia articles.

5 The Disambiguation Component

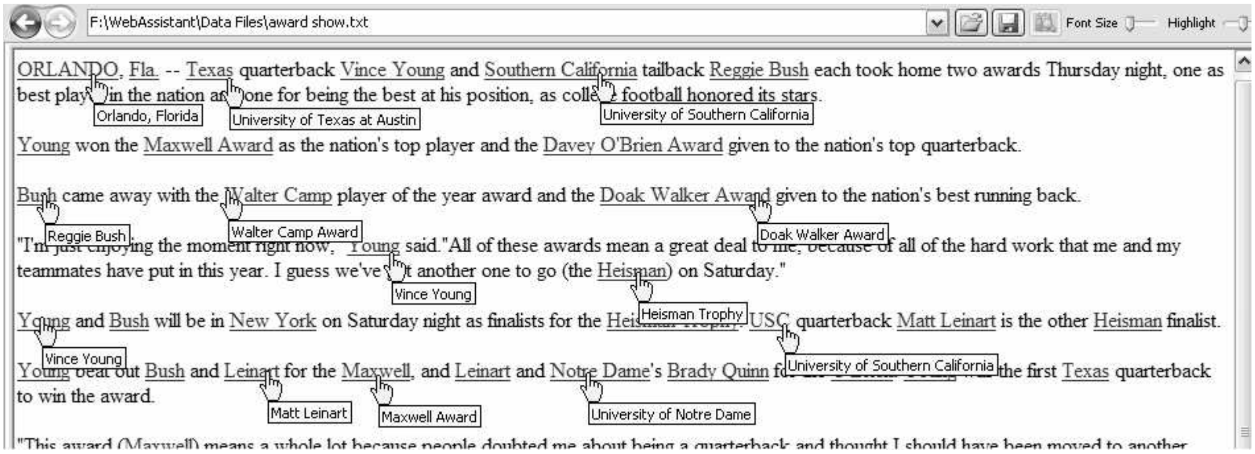
The disambiguation process employs a vector space model, in which a vectorial representation of the processed document is compared with the vectorial representations of the Wikipedia entities.

Once the named entity surface forms were identified and the in-document coreferences hypothesized, the system retrieves all possible entity disambiguations of each surface form. Their Wikipedia contexts that occur in the document and their category tags are aggregated into a *document vector*, which is subsequently compared with the Wikipedia *entity vector* (of categories and contexts) of each possible entity disambiguation. We then choose the assignment of entities to surface forms that maximizes the similarity between the document vector and the entity vectors, as we explain further.

Formally, let $C = \{c_1,\ldots,c_M\}$ be the set of known contexts from Wikipedia and $\mathcal{T} = \{t_1,\ldots,t_N\}$ the set of known category tags. An entity e can then be represented as a vector $\delta_e \in \{0,1\}^{M+N}$, with two components, $\delta_e|_C \in \{0,1\}^M$ and $\delta_e|_{\mathcal{T}} \in \{0,1\}^N$, corresponding to the context and category information, respectively:

$$\delta_e^{\,i} = \begin{cases} 1, & \text{if } c_i \text{ is a context for entity } e \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_e^{\,M+j} = \begin{cases} 1, & \text{if } t_j \text{ is a category tag for } e \\ 0, & \text{otherwise.} \end{cases}$$



ORLANDO, Fla. -- Texas quarterback Vince Young and Southern California tailback Reggie Bush each took home two awards Thursday night, one as best play in the nation a one for being the best at his position, as coll football honored its stars.

Young won the Maxwell Award as the nation's top player and the Davey O'Brien Award given to the nation's top quarterback.

Bush came away with the Walter Camp player of the year award and the Doak Walker Award given to the nation's best running back.

"I'm just enjoying the moment right now," Young said."All of these awards mean a great deal to me, because of all of the hard work that me and my teammates have put in this year. I guess we've got another one to go (the Heisman) on Saturday."

Young and Bush will be in New York on Saturday night as finalists for the Heisman Trophy. USC quarterback Matt Leinart is the other Heisman finalist.

Young beat out Bush and Leinart for the Maxwell, and Leinart and Notre Dame's Brady Quinn for the O'Brien. Young was the first Texas quarterback to win the award.

"This award (Maxwell) means a whole lot because people doubted me about being a quarterback and thought I should have been moved to another

Figure 3. Screenshot of the implemented system showing an example of analyzed text. The superimposed tooltips show how several of the surface forms were disambiguated based on the context and category agreement method.

Let ε(*s*) denote the set of entities that are known to have a surface form *s*. For example, recalling Figure 1, Colombia (the country) and Columbia University are entities that are known to have the surface form "Columbia". Let *D* be the analyzed document and $S(D) = \{s_1,\ldots,s_n\}$ the set of surface forms identified in *D*. We build its context vector $d = \{d_1,\ldots,d_M\} \in \mathbb{N}^M$, where d_i is the number of occurrences of context c_i in *D*. To account for all possible disambiguations of the surface forms in *D*, we also build an extended vector $\bar{d} \in \mathbb{N}^{M+N}$ so that $\bar{d}\,|_C = d$ and $\bar{d}\,|_T = \sum_{s\in S(D)} \sum_{e\in\varepsilon(s)} \delta_e\,|_T$.[2]

Our goal is to find the assignment of entities to surface forms $s_i \mapsto e_i$, $i \in 1..n$, that maximizes the agreement between $\delta_{ei}|_C$ and *d*, as well as the agreement between the categories of any two entities $\delta_{e_i}|_T$ and $\delta_{e_j}|_T$. This can be written as:

$$\underset{\substack{(e_1,..,e_n)\in \\ \varepsilon(s_1)\times.\times\varepsilon(s_n)}}{\arg\max} \sum_{i=1}^{n} < \delta_{e_i}|_C, d > + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} < \delta_{e_i}|_T, \delta_{e_j}|_T > , \quad (1)$$

where $<\cdot,\cdot>$ denotes the scalar product of vectors. Note that the quality of an assignment of an entity to a surface form depends on all the other assignments made, which makes this a difficult optimization problem. An arguably more robust strategy to account for category agreement, which also proves to be computationally efficient, is to maximize the agreement between the categories of the assigned entity to each surface form and all possible disambiguations of the other surface forms in *D*. We will show that this is equivalent to computing:

$$\underset{(e_1,..,e_n)\in\varepsilon(s_1)\times..\times\varepsilon(s_n)}{\arg\max} \sum_{i=1}^{n} < \delta_{e_i}, \bar{d} - \delta_{e_i}|_T > \qquad (2)$$

Indeed, using the definition of $\bar{d}$ and partitioning the context and category components, we can rewrite the sum in equation (2) as

$$\sum_{i=1}^{n} < \delta_{e_i}|_C, d > + \sum_{i=1}^{n} < \delta_{e_i}|_T, \bar{d}|_T - \delta_{e_i}|_T > =$$

$$\sum_{i=1}^{n} < \delta_{e_i}|_C, d > + \sum_{i=1}^{n} < \delta_{e_i}|_T, \Big(\sum_{j=1}^{n} \sum_{e\in\varepsilon(s_j)} \delta_e|_T \Big) - \delta_{e_i}|_T > =$$

$$\sum_{i=1}^{n} < \delta_{e_i}|_C, d > + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} < \delta_{e_i}|_T, \sum_{e\in\varepsilon(s_j)} \delta_e|_T > \quad \text{(q.e.d.)}$$

Note now that the maximization of the sum in (2) is equivalent to the maximization of each of its terms, which means that the computation reduces to $\underset{e_i\in\varepsilon(s_i)}{\arg\max} < \delta_{e_i}, \bar{d} - \delta_{e_i}|_T >, i \in 1..n$., or equivalently,

$$\underset{e_i\in\varepsilon(s_i)}{\arg\max} < \delta_{e_i}, \bar{d} > - \| \delta_{e_i}|_T \|^2, i \in 1..n \qquad (3)$$

Our disambiguation process therefore employs two steps: first, it builds the extended document vector and second, it maximizes the scalar products in equation (3). In practice, it is not necessary to build the document vector over all contexts *C*, but only over the contexts of the possible entity disambiguations of the surface forms in the document.

Also note that we are not normalizing the scalar products by the norms of the vectors (which would lead to the computation of cosine similarity). In this manner, we implicitly account for the frequency with which a surface form is used to mention various entities and for the importance of these entities (important entities have longer Wikipedia articles, are mentioned more frequently in other articles, and also tend to have more category tags).

While rarely, one surface form can be used to mention two or more different entities in a document (e.g., "Supreme Court" may refer to the federal institution in one paragraph and to a state's judicial institution in another paragraph). To account for such cases, the described disambiguation process is performed iteratively for the instances of the surface forms with multiple disambiguations with similarity scores higher than an empirically determined threshold, by shrinking the context used for the disambiguation of each instance from document level to paragraph level, and if necessary, to sentence level.

6 Evaluation

We used as development data for building the described system the Wikipedia collection as of April 2, 2006 and a set of 100 news stories on a diverse range of topics. For the final evaluation, we performed data extraction from the September 11, 2006 version of the Wikipedia collection.

We evaluated the system in two ways: on a set of Wikipedia articles, by comparing the system output with the references created by human contributors, and on a set of news stories, by doing a post-hoc evaluation of the system output. The evaluation data can be downloaded from http://research.microsoft.com/users/silviu/WebAssistant/TestData.

[2] We use the notation $\bar{d}$ to emphasize that this vector contains information that was not present in the original document *D*.

In both settings, we computed a disambiguation baseline in the following manner: for each surface form, if there was an entity page or redirect page whose title matches exactly the surface form then we chose the corresponding entity as the baseline disambiguation; otherwise, we chose the entity most frequently mentioned in Wikipedia using that surface form.

6.1 Wikipedia Articles

We selected at random 350 Wikipedia entity pages and we discarded their content during the information extraction phase. We then performed an automatic evaluation, in which we compared the hyperlinks created by our system with the links created by the Wikipedia contributors. In an attempt to discard most of the non-named entities, we only kept for evaluation the surface forms that started with an uppercase letter. The test articles contained 5,812 such surface forms. 551 of them referenced non-existing articles (for example, the filmography section of a director contained linked mentions of all his movies although many of them did not have an associated Wikipedia page). Also, 130 of the surface forms were not used in other Wikipedia articles and therefore both the baseline and the proposed system could not hypothesize a disambiguation for them. The accuracy on the remaining 5,131 surface forms was 86.2% for the baseline system and **88.3%** for the proposed system. A McNemar test showed that the difference is not significant, the main cause being that the majority of the test surface forms were unambiguous. When restricting the test set only to the 1,668 ambiguous surface forms, the difference in accuracy between the two systems is significant at $p = 0.01$. An error analysis showed that the Wikipedia set used as gold standard contained relatively many surface forms with erroneous or out-of-date links, many of them being correctly disambiguated by the proposed system (thus, counted as errors). For example, the test page “The Gods (band)” links to Paul Newton, the painter, and Uriah Heep, which is a disambiguation page, probably because the original pages changed over time, while the proposed system correctly hypothesizes links to Paul Newton (musician) and Uriah Heep (band).

6.2 News Stories

We downloaded the top two stories in the ten MSNBC news categories (Business, U.S. Politics, Entertainment, Health, Sports, Tech & Science, Travel, TV News, U.S. News, and World News) as of January 2, 2007 and we used them as input to our system. We then performed a post-hoc evaluation of the disambiguations hypothesized for the surface forms correctly identified by the system (i.e. if the boundaries of a surface form were not identified correctly then we disregarded it).

We defined a disambiguation to be correct if it represented the best possible Wikipedia article that would satisfy a user’s need for information and incorrect otherwise. For example, the article Viking program is judged as correct for “Viking Landers”, for which there is no separate article in the Wikipedia collection. Linking a surface form to a wrong Wikipedia article was counted as an error regardless whether or not an appropriate Wikipedia article existed. When the system could not disambiguate a surface form (e.g. “N’ Sync”, “’Bama”, and “Harris County Jail”), we performed a search in Wikipedia for the appropriate entity. If an article for that entity existed (e.g., ’N Sync and Alabama) then we counted that instance as an error. Otherwise, we counted it separately as non-recallable (e.g. there is no Wikipedia article for the Harris County Jail entity and the article for Harris County, Texas does not discuss the jail system).

The test set contained 756 surface forms, of which 127 were non-recallable. The proposed system obtained an accuracy of **91.4%**, versus a 51.7% baseline (significant at $p = 0.01$). An analysis of these data showed not only that the most common surface forms used in news are highly ambiguous but also that a large number of Wikipedia pages with titles that are popular surface forms in news discuss subjects different from those with common news usage (e.g., the page titled “China” discusses the Chinese civilization and is not the correct assignment for the People's Republic of China entity; similarly, the default page for “Blackberry” talks about the fruit rather than the wireless company with the same name).

7 Conclusions and Potential Impact

We presented a large scale named entity disambiguation system that employs a huge amount of information automatically extracted from Wikipedia over a space of more than 1.4 million entities. In tests on both real news data and Wikipedia text, the system obtained accuracies exceeding 91% and 88%. Because the entity recognition and disambiguation processes employed use very little language-dependent resources additional to Wikipedia, the system can be easily adapted to languages other than English.

The system described in this paper has been fully implemented as a Web browser (Figure 3), which can analyze any Web page or client text document. The application on a large scale of such an entity extraction and disambiguation system could result in a move from the current space of words to a space of concepts, which enables several paradigm shifts and opens new research directions, which we are currently investigating, from entity-based indexing and searching of document collections to personalized views of the Web through entity-based user bookmarks.

Acknowledgments

The author would like to gratefully thank Mike Schultz and Robert Ragno for their help in building the system infrastructure, Microsoft Live Search for providing access to their search engine, and the anonymous reviewers for their useful comments.

References

Bagga, A. and B. Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of COLING-ACL*, 79-85.

Bunescu, R. and M. Paşca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proceedings of EACL*, 9-16.

Cederberg, S. and D. Widdows. 2003. Using LSA and noun coordination information to improve the precision and recall of hyponymy extraction. In *Proceedings of CoNLL*, 111-118.

Doddington, G., A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. ACE program – task definitions and performance measures. In *Proceedings of LREC*, 837-840.

Edmonds, P. and S. Cotton. 2001. Senseval-2 overview. In *Proceedings of SENSEVAL-2*, 1-6.

Gabrilovich, E. and S. Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. *Proceedings of IJCAI*, 1606-1611.

Gale, W., K. Church, and D. Yarowsky. 1992. One sense per discourse. In *Proceedings of the 4th DARPA SNL Workshop*, 233-237.

Grishman, R. and B. Sundheim. 1996. Message Understanding Conference - 6: A brief history. In *Proceedings of COLING*, 466-471.

Hearst, M. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proc. COLING*, 539-545.

Hirschman, L. and N. Chinchor. 1997. MUC-7 Coreference Task Definition. In *Proceedings of MUC-7*.

Kanada, Y. 1999. A method of geographical name extraction from Japanese text. In *Proceedings of CIKM*, 46-54.

Kilgarriff, A. and J. Rosenzweig. 2000. Framework and results for English Senseval. *Computers and Humanities, Special Issue on SENSEVAL*, 15-48.

Lapata, M. and F. Keller. 2004. The Web as a Baseline: Evaluating the Performance of Unsupervised Web-based Models for a Range of NLP Tasks. In *Proceedings of HLT*, 121-128.

Mann, G. S. and D. Yarowsky. 2003. Unsupervised Personal Name Disambiguation. In *Proceedings of CoNLL*, 33-40.

Mihalcea, R., T. Chklovski, and A. Kilgarriff. The Senseval-3 English lexical sample task. In *Proceedings of SENSEVAL-3*, 25-28.

Overell, S., and S. Rüger. 2006 Identifying and grounding descriptions of places. In *SIGIR Workshop on Geographic Information Retrieval*.

Raghavan, H., J. Allan, and A. McCallum. 2004. An exploration of entity models, collective classification and relation description. In *KDD Workshop on Link Analysis and Group Detection*.

Ravin, Y. and Z. Kazi. 1999. Is Hillary Rodham Clinton the President? In *ACL Workshop on Coreference and it's Applications*.

Remy, M. 2002. Wikipedia: The free encyclopedia. In *Online Information Review*, 26(6): 434.

Roark, B. and E. Charniak. 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of COLING-ACL*, 1110-1116.

Salton, G. 1989. *Automatic Text Processing*. Addison-Wesley.

Smith, D. A. and G. Crane. 2002. Disambiguating geographic names in a historic digital library. In *Proceedings of ECDL*, 127-136.

Strube, M. and S. P. Ponzeto. 2006. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of AAAI*, 1419-1424.

Tjong Kim Sang, E. F. and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL*, 142-147.

Wacholder, N., Y. Ravin, and M. Choi. 1997. Disambiguation of proper names in text. In *Proceedings of ANLP*, 202-208.

Woodruff, A. G. and C. Paunt. GIPSY:Automatic geographic indexing of documents. *Journal of the American Society for Information Science and Technology*, 45(9):645-655.

Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions

Siddharth Patwardhan and **Ellen Riloff**
School of Computing
University of Utah
Salt Lake City, UT 84112
{sidd,riloff}@cs.utah.edu

Abstract

We present an information extraction system that decouples the tasks of finding relevant regions of text and applying extraction patterns. We create a self-trained relevant sentence classifier to identify relevant regions, and use a *semantic affinity* measure to automatically learn domain-relevant extraction patterns. We then distinguish *primary* patterns from *secondary* patterns and apply the patterns selectively in the relevant regions. The resulting IE system achieves good performance on the MUC-4 terrorism corpus and ProMed disease outbreak stories. This approach requires only a few seed extraction patterns and a collection of relevant and irrelevant documents for training.

1 Introduction

Many information extraction (IE) systems rely on rules or patterns to extract words and phrases based on their surrounding context (e.g., (Soderland et al., 1995; Riloff, 1996; Califf and Mooney, 1999; Yangarber et al., 2000)). For example, a pattern like *"<subject> was assassinated"* can reliably identify a victim of a murder event. Classification-based IE systems (e.g., (Freitag, 1998; Freitag and McCallum, 2000; Chieu et al., 2003)) also generally decide whether to extract words based on properties of the words themselves as well as properties associated with their surrounding context.

In this research, we propose an alternative approach to IE that decouples the tasks of finding a relevant region of text and finding a desired extraction. In a typical pattern-based IE system, the extraction patterns perform two tasks: (a) they recognize that a relevant incident has occurred, and (b) they identify and extract some information about that event. In contrast, our approach first identifies relevant regions of a document that describes relevant events, and then applies extraction patterns only in these relevant regions.

This decoupled approach to IE has several potential advantages. First, even seemingly good patterns can produce false hits due to metaphor and idiomatic expressions. However, by restricting their use to relevant regions of text, we could avoid such false positives. For example, *"John Kerry attacked George Bush"* is a metaphorical description of a verbal tirade, but could be easily mistaken for a physical attack. Second, IE systems are prone to errors of omission when relevant information is not explicitly linked to an event. For instance, a phrase like *"the gun was found..."* does not directly state that the the gun was used in a terrorist attack. But if the gun is mentioned in a region that clearly describes a terrorist attack, then it can be reasonably inferred to have been used in the attack. Third, if the IE patterns are restricted to areas of text that are known to be relevant, then it may suffice to use relatively general patterns, which may be easier to learn or acquire.

Our approach begins with a relevant sentence classifier that is trained using only a few seed patterns and a set of relevant and irrelevant documents (but no sentence-level annotations) for the domain of interest. The classifier is then responsible for identifying sentences that are relevant to the IE task. Next, we learn "semantically appropriate" extraction pat-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 717–727, Prague, June 2007. ©2007 Association for Computational Linguistics

terns by evaluating candidate patterns using a *semantic affinity* metric. We then separate the patterns into *primary* and *secondary* patterns, and apply them selectively to sentences based on the relevance judgments produced by the classifier. We evaluate our IE system on two data sets: the MUC-4 IE terrorism corpus and ProMed disease outbreak articles. Our results show that this approach works well, often outperforming the AutoSlog-TS IE system which benefits from human review.

2 Motivation and Related Work

Our research focuses on event-oriented information extraction (IE), where the goal of the IE system is to extract facts associated with domain-specific events from unstructured text. Many different approaches to information extraction have been developed, but generally speaking they fall into two categories: classifier-based approaches and rule/pattern-based approaches.

Classifier-based IE systems use machine learning techniques to train a classifier that sequentially processes a document looking for words to be extracted. Examples of classifier-based IE systems are SRV (Freitag, 1998), HMM approaches (Freitag and McCallum, 2000), ALICE (Chieu et al., 2003), and Relational Markov Networks (Bunescu and Mooney, 2004). The classifier typically decides whether a word should be extracted by considering features associated with that word as well as features of the words around it.

Another common approach to information extraction uses a set of explicit patterns or rules to find relevant information. Some older systems relied on hand-crafted patterns, while more recent systems learn them automatically or semi-automatically. Examples of rule/pattern-based approaches to information extraction are FASTUS (Hobbs et al., 1997), PALKA (Kim and Moldovan, 1993), LIEP (Huffman, 1996), CRYSTAL (Soderland et al., 1995), AutoSlog/AutoSlog-TS (Riloff, 1993; Riloff, 1996), RAPIER (Califf and Mooney, 1999), WHISK (Soderland, 1999), ExDisco (Yangarber et al., 2000), SNOWBALL (Agichtein and Gravano, 2000), $(LP)^2$ (Ciravegna, 2001), subtree patterns (Sudo et al., 2003), predicate-argument rules (Yakushiji et al., 2006) and KnowItAll (Popescu et al., 2004).

One commonality behind all of these approaches is that they simultaneously decide whether a context is relevant and whether a word or phrase is a desirable extraction. Classifier-based systems rely on features that consider both the word and its surrounding context, and rule/pattern-based systems typically use patterns or rules that match both the words around a candidate extraction and (sometimes) properties of the candidate extraction itself.

There is a simplicity and elegance to having a single model that handles both of these problems at the same time, but we hypothesized that there may be benefits to decoupling these tasks. We investigate an alternative approach that involves two passes over a document. In the first pass, we apply a *relevant region identifier* to identify regions of the text that appear to be especially relevant to the domain of interest. In the second pass, we apply extraction patterns inside the relevant regions. We hypothesize three possible benefits of this decoupled approach.

First, if a system is certain that a region is relevant, then it can be more aggressive about searching for extractions. For example, consider the domain of terrorist event reports, where a goal is to identify the weapons that were used. Existing systems generally require rules/patterns to recognize a context in which a weapon is explicitly linked to an event or its consequences (e.g., "*attack with <np>*", or "*<np> caused damage*"). However, weapons are not always directly linked to an event in text, but they may be inferred through context. For instance, an article may mention that a weapon was "found" or "used" without explicitly stating that it was involved in a terrorist event. However, if we know in advance that we are in a relevant context, then we can reliably infer that the weapon was, most likely, used in the event.

Second, some patterns may seem to be relevant locally, but they can be deemed irrelevant when the global context is considered. For example, consider these sentences from the MUC-4 terrorism corpus:

> *D'Aubuisson unleashed harsh attacks on Duarte ...*
> *Other brave minds that advocated reform had been killed before in that struggle.*

Locally, patterns such as "*<subject> unleashed* *attacks"* and *"<subject> had been killed"* seem likely to identify the perpetrators and victims of a physical attack. But when read in the full context of these sentences, it becomes clear that they are not related to a specific physical attack.

Third, decoupling these tasks may simplify the learning process. Identifying relevant regions amounts to a text classification task, albeit the goal is to identify not just relevant documents, but relevant sub-regions of documents. Within a relevant region the patterns may not need to be as discriminating. So a more general learning approach may suffice.

In this paper, we describe an IE system that consists of two decoupled modules for relevant sentence identification and extraction pattern learning. In Section 3, we describe the self-trained sentence classifier, which requires only a few seed patterns and relevant and irrelevant documents for training. Section 4 describes the extraction pattern learning module, which identifies semantically appropriate patterns for the IE system using a *semantic affinity* measure. Section 5 explains how we distinguish Primary patterns from Secondary patterns. Section 6 presents experimental results on two domains. Finally, Section 7 lists our conclusions and future work.

3 A Self-Trained Relevant Sentence Classifier

Our hypothesis is that if a system can reliably identify relevant regions of text, then extracting information only from these relevant regions can improve IE performance. There are many possible definitions for *relevant region* (e.g., Salton et al. (1993), Callan (1994)), and exploring the range of possibilities is an interesting avenue for future work. For our initial investigations of this idea, we begin by simply defining a sentence as our region size. This has the advantage of being an easy boundary line to draw (i.e., it is relatively easy to identify sentence boundaries) and it is a small region size yet includes more context than most current IE systems do[1].

Our goal is to create a classifier that can determine whether a sentence contains information that should be extracted. Furthermore, we wanted to create a classifier that does not depend on manually annotated sentence data so that our system can be easily ported across domains. Therefore, we devised a method to self-train a classifier using a training set of relevant and irrelevant documents for the domain, and a few seed patterns as input. However, this results in an asymmetry in the training set. By definition, if a document is irrelevant to the IE task, then it cannot contain any relevant information. Consequently, *all sentences in an irrelevant document must be irrelevant*, so these sentences form our initial *irrelevant sentences pool*. In contrast, if a document is relevant to the IE task, then there must be at least one sentence that contains relevant information. However, most documents contain a mix of both relevant and irrelevant sentences. Therefore, the sentences from the relevant documents form our *unlabeled sentences pool*.

Figure 1 shows the self-training procedure, which begins with a handful of *seed patterns* to initiate the learning process. The seed patterns should be able to reliably identify some information that is relevant to the IE task. For instance, to build an IE system for terrorist incident reports, we used seed patterns such as *"<subject> was kidnapped"* and *"assassination of <np>"*. The patterns serve as a simple pattern-based classifier to automatically identify some relevant sentences. In *iteration 0* of the self-training loop (shown as dotted lines in Figure 1), the pattern-based classifier is applied to the unlabeled sentences to automatically label some of them as relevant.

Next, an SVM (Vapnik, 1995) classifier[2] is trained using these relevant sentences and an equal number of irrelevant sentences randomly drawn from the irrelevant sentences pool. We artificially created a balanced training set because the set of irrelevant sentences is initially much larger than the set of relevant sentences, and we want the classifier to learn how to identify new relevant sentences. The feature set consists of all unigrams that appear in the training set. The SVM is trained using a linear kernel with the default parameter settings. In a self-training loop, the classifier is then applied to the unlabeled sentences, and all sentences that it classifies as relevant are added to the relevant sentences pool. The classifier is then retrained with all of the

[1] Most IE systems only consider a context window consisting of a few words or phrases on either side of a potential extraction.

[2] We used the freely available SVM^{light} (Joachims, 1998) implementation: http://svmlight.joachims.org

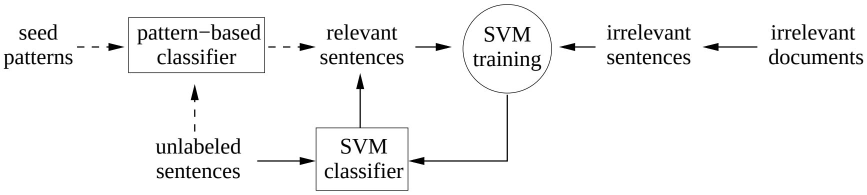


Figure 1: The Training Process to Create a Relevant Sentence Classifier

relevant sentences and an equal number of irrelevant sentences, and the process repeats. We ran this self-training procedure for three iterations and then used the resulting classifier as our *relevant sentence classifier* in the IE experiments described in Section 6.3.

4 Learning Semantic Affinity-based Extraction Patterns

One motivation for creating a relevant region classifier is to reduce the responsibilities of the extraction patterns. Once we know that we are in a domain-relevant area of text, patterns that simply identify words and phrases belonging to a relevant semantic class may be sufficient. In this section, we describe a method to automatically identify semantically appropriate extraction patterns for use with the sentence classifier.

In previous work (Patwardhan and Riloff, 2006), we introduced a metric called *semantic affinity* which was used to automatically assign event roles to extraction patterns. Semantic affinity measures the tendency of a pattern to extract noun phrases that belong to a specific set of semantic categories. To use this metric for information extraction, a mapping must be defined between semantic categories and the event roles that are relevant to the IE task. For example, one role in the terrorism domain is *physical target*, which refers to physical objects that are the target of an attack. Most physical targets fall into one of two general semantic categories: BUILDING or VEHICLE. Consequently, we define the mapping "Target → BUILDING, VEHICLE". Similarly, we might define the mapping "Victim → HUMAN, ANIMAL, PLANT" to characterize possible victims of disease outbreaks. Each semantic category must be mapped to a single event role. This is a limitation of our approach for domains where multiple roles can be filled by the same class of fillers. However, sometimes a general semantic class can be partitioned into subclasses that are associated with different roles. For example, in the terrorism domain, both perpetrators and victims belong to the general semantic class HUMAN. But we used the subclasses TERRORIST-HUMAN, which represents likely perpetrator words (e.g., "terrorist", "guerrilla", and "gunman") and CIVILIAN-HUMAN, which represents ordinary people (e.g., "photographer","rancher", and "tourist"), in order to generate different semantic affinity estimates for the perpetrator and victim roles.

To determine the semantic category of a noun, we use the Sundance parser (Riloff and Phillips, 2004), which contains a dictionary of words that have semantic category labels. Alternatively, a resource such as WordNet (Fellbaum, 1998) could be used to obtain this information. All semantic categories that cannot be mapped to a relevant event role are mapped to a special Other role.

To estimate the semantic affinity of a pattern p for an event role r_k, the system computes $\mathrm{f}(p, r_k)$, which is the number of pattern p's extractions that have a head noun belonging to a semantic category mapped to r_k. These frequency counts are obtained by applying each pattern to the training corpus and collecting its extractions. The *semantic affinity* of a pattern p with respect to an event role r_k is formally defined as:

$$\mathrm{sem\_aff}(p, r_k) = \frac{\mathrm{f}(p, r_k)}{\sum_{i=1}^{|R|} \mathrm{f}(p, r_i)} \log_2 \mathrm{f}(p, r_k) \quad (1)$$

where R is the set of event roles $\{r_1, r_2, \ldots, r_{|R|}\}$. Semantic affinity is essentially the probability that a phrase extracted by pattern p will be a semantically appropriate filler for role r_k, weighted by the log of the frequency.[3] Note that it is possible for a

[3] This formula is very similar to pattern ranking metrics used by previous IE systems (Riloff, 1996; Yangarber et al., 2000), although not for semantics.

pattern to have a semantic affinity for multiple event roles. For instance, a terrorism pattern like *"attack on <np>"* may have a semantic affinity for both Targets and Victims.

To generate extraction patterns for an IE task, we first apply the AutoSlog (Riloff, 1993) extraction pattern generator to the training corpus exhaustively, so that it literally generates a pattern to extract every noun phrase in the corpus. Then for each event role, we rank the patterns based on their semantic affinity for that role.

Figure 2 shows the 10 patterns with the highest semantic affinity scores for 4 event roles. In the terrorism domain, we show patterns that extract *weapons* and *perpetrator organizations* (PerpOrg). In the disease outbreaks domain, we show patterns that extract *diseases* and *victims*. The patterns rely on shallow parsing, syntactic role assignment (e.g., subject (*subject*) and direct object (*dobj*) identification), and active/passive voice recognition, but they are shown here in a simplified form for readability. The portion in brackets (between < and >) is extracted, and the other words must match the surrounding context. In some cases, all of the matched words are extracted (e.g., "<# birds>"). Most of the highest-ranked victim patterns recognize noun phrases that refer to people or animals because they are common in the disease outbreak stories and these patterns do not extract information that is associated with any competing event roles.

Top Terrorism Patterns	
Weapon	PerpOrg
<subject> exploded	<subject> claimed
planted <dobj>	panama from <np>
fired <dobj>	<np> claimed responsibility
<subject> was planted	command of <np>
explosion of <np>	wing of <np>
<subject> was detonated	kidnapped by <np>
<subject> was set off	guerillas of <np>
set off <dobj>	<subject> operating
hurled <dobj>	kingpins of <np>
<subject> was placed	attacks by <np>

Top Disease Outbreak Patterns	
Disease	Victim
cases of <np>	<# people>
spread of <np>	<# cases>
outbreak of <np>	<# birds>
<#th outbreak>	<# animals>
<# outbreaks>	<subject> died
case of <np>	<# crows>
contracted <dobj>	<subject> know
outbreaks of <np>	<# pigs>
<# viruses>	<# cattle>
spread of <np>	<# sheep>

Figure 2: Top-Ranked Extraction Patterns

5 Distinguishing Primary and Secondary Patterns

So far, our goal has been to find relevant areas of text, and then apply semantically appropriate patterns in those regions. Our expectation was that fairly general, semantically appropriate patterns could be effective if their range is restricted to regions that are known to be relevant. If our relevant sentence classifier was perfect, then performing IE only on relevant regions would be ideal. However, identifying relevant regions is a difficult problem in its own right, and our relevant sentence classifier is far from perfect.

Consequently, one limitation of our proposed approach is that no IE would be performed in sentences that are not deemed to be relevant by the classifier, and this could negatively affect recall. We addressed this issue by allowing reliable patterns to be applied to all sentences in the text, irrespective of the output of the sentence classifier. For example, the pattern *"<subject> was assassinated"* is a clear indicator of a murder event, and does not need to be restricted by the sentence classifier[4]. We will refer to such reliable patterns as *Primary Patterns*. In contrast, patterns that are not necessarily reliable and need to be restricted to relevant regions will be called *Secondary Patterns*.

To automatically distinguish Primary Patterns from Secondary Patterns, we compute the conditional probability of a pattern p being relevant, $\Pr(relevant \mid p)$, based on the relevant and irrelevant documents in our training set. We then define an upper conditional probability threshold θ_u to separate Primary patterns from Secondary Patterns. If a pattern has a high correlation with relevant documents, then our assumption is that it is generally a reliable pattern that is not likely to occur in irrelevant contexts.

On the flip side, we can also use this conditional probability to weed out patterns that rarely

[4] In other words, if such a pattern matches a sentence that is classified as irrelevant, then the classifier is probably incorrect.

appear in relevant documents. Such patterns (e.g., "<*subject*> *held*", "<*subject*> *saw*", etc.) could potentially have a high semantic affinity for one of the semantic categories, but they are not likely to be useful if they mainly occur in irrelevant documents. As a result, we also define a lower conditional probability threshold θ_l that identifies irrelevant extraction patterns.

The two thresholds θ_u and θ_l are used with semantic affinity to identify the most appropriate Primary and Secondary patterns for the task. This is done by first removing from our extraction pattern collection all patterns with probability less than θ_l. For each event role, we then sort the remaining patterns based on their semantic affinity score for that role and select the top N patterns. Next, we use the θ_u probability threshold to separate these N patterns into two subsets. Patterns with a probability above θ_u are considered to be Primary patterns for that role, and those below become the Secondary patterns.

6 Experiments and Results

6.1 Data Sets

We evaluated the performance of our IE system on two data sets: the MUC-4 terrorism corpus (Sundheim, 1992), and a ProMed disease outbreaks corpus. The MUC-4 IE task is to extract information about Latin American terrorist events. We focused our analysis on five MUC-4 string roles: *perpetrator individuals*, *perpetrator organizations*, *physical targets*, *victims*, and *weapons*. The disease outbreaks corpus consists of electronic reports about disease outbreak events. For this domain we focused on two string roles: *diseases* and *victims*[5].

The MUC-4 data set consists of 1700 documents, divided into 1300 development (DEV) texts, and four test sets of 100 texts each (TST1, TST2, TST3, and TST4). We used 1300 texts (DEV) as our training set, 200 texts (TST1+TST2) for tuning, and 200 texts (TST3+TST4) as a test set. All 1700 documents have answer key templates. For the training set, we used the answer keys to separate the documents into relevant and irrelevant subsets. Any document containing at least one relevant event was considered relevant.

For the disease outbreak domain the data set was collected from ProMed-mail[6], an open-source, global electronic reporting system for outbreaks of infectious diseases. We collected thousands of ProMed reports and created answer key templates for 245 randomly selected articles. We used 125 as a tuning set, and 120 as the test set. We used 2000 different documents as the relevant documents for training. Most of the ProMed articles contain email headers, footers, citations, and other snippets of non-narrative text, so we wrote a "zoner" program[7] to automatically strip off some of this extraneous information.

To obtain irrelevant documents, we collected 4000 biomedical abstracts from PubMed[8], a free archive of biomedical literature. We collected twice as many irrelevant documents because the PubMed articles are roughly half the size of the ProMed articles, on average. To ensure that the PubMed articles were truly irrelevant (i.e. did not contain any disease outbreak reports) we used specific queries to exclude disease outbreak abstracts.

The complete IE task involves the creation of answer key templates, one template per incident[9]. Template generation is a complex process, requiring coreference resolution and discourse analysis to determine how many incidents were reported and which facts belong with each incident. Our work focuses on extraction pattern learning and not template generation, so we evaluated our systems directly on the extractions themselves, before template generation would take place. This approach directly measures how accurately the patterns find relevant information, without confounding factors from the template generation process. For example, if a coreference resolver incorrectly decides that two extractions are coreferent and merges them, then only one extraction would be scored. We used a *head noun* scoring scheme, where an extraction is considered to be correct if its head noun matches the head noun in the answer key[10]. Also, pronouns were discarded from both the system responses and the answer keys since no coreference resolution is done. Duplicate

[5] The "victims" can be people, animals, or plants that are affected by a disease.

[6] http://www.promedmail.org

[7] The term *zoner* was introduced by Yangarber et al. (2002).

[8] http://www.pubmedcentral.nih.gov

[9] Many of the stories have multiple incidents per article.

[10] For example, "armed men" will match "5 armed men".

extractions (e.g., the same string extracted by different patterns) were conflated before being scored, so they count as just one hit or one miss.

6.2 Relevant Sentence Classifier Results

First, we evaluated the performance of the relevant sentence classifier described in Section 3. We automatically generated seed patterns from the training texts. AutoSlog (Riloff, 1993) was used to generate all extraction patterns that appear in the training documents, and only those patterns with frequency $>$ 50 were kept. These were then ranked by $\Pr(relevant \mid p)$, and the top 20 patterns were chosen as seeds. In the disease outbreak domain, 54 patterns had a frequency $>$ 50 and probability of 1.0. We wanted to use the same number of seeds in both domains for consistency, so we manually reviewed them and used the 20 most domain-specific patterns as seeds.

Due to the greater stylistic differences between the relevant and irrelevant documents in the disease outbreak domain (since they were gathered from different sources), we decided to make the classifier for that domain more conservative in classifying documents as relevant. To do this we used the prediction scores output by the SVM as a measure of confidence in the classification. These scores are essentially the distance of the test examples from the support vectors of the SVM. For the disease outbreaks domain we used a cutoff of 1.0 and in the terrorism domain we used the default of 0.

Since we do not have sentence annotated data, there is no direct way to evaluate the classifiers. However, we did an indirect evaluation by using the answer keys from the tuning set. If a sentence in a tuning document contained a string that occurred in the corresponding answer key template, then we considered that sentence to be relevant. Otherwise, the sentence was deemed irrelevant. This evaluation is not perfect for two reasons: (1) answer key strings do not always appear in relevant sentences.[11], and (2) some arguably relevant sentences may not contain an answer key string (e.g., they may contain a pronoun that refers to the answer, but the pronoun itself is not the desired extraction). However, judging the relevance of sentences without relying on answer keys is also tricky, so we decided that this approach was probably good enough to get a reasonable assessment of the classifier. Using this criterion, 17% of the sentences in the terrorism articles are relevant, and 28% of the sentences in the disease outbreaks articles are relevant.

		Irrelevant			Relevant		
	Acc	Rec	Pr	F	Rec	Pr	F
Terrorism							
Iter #1	.84	.93	.89	.91	.41	.55	.47
Iter #2	.84	.90	.91	.90	.54	.51	.53
Iter #3	.82	.85	.92	.89	.63	.46	.53
Disease Outbreaks							
Iter #1	.75	.96	.76	.85	.21	.66	.32
Iter #2	.71	.76	.82	.79	.58	.48	.53
Iter #3	.63	.60	.85	.70	.72	.41	.52

Table 1: Relevant Sentence Classifier Evaluation

Table 1 shows the accuracy, recall, precision, and F scores of the SVM classifiers after each self-training iteration. The classifiers generated after the third iteration were used in our IE experiments. The final accuracy is 82% in the terrorism domain, and 63% for the disease outbreaks domain. The precision on irrelevant sentences is high in both domains, but the precision on relevant sentences is relatively weak. Despite this, we will show in Section 6.3 that the classifier is effective for the IE task. The reason why the classifier improves IE performance is because it favorably alters the proportion of relevant sentences that are passed along to the IE system. For example, an analysis of the tuning set shows that removing the sentences deemed to be irrelevant by the classifier increases the proportion of relevant sentences from 17% to 46% in the terrorism domain, and from 28% to 41% in the disease outbreaks domain.

We will also see in Section 6.3 that IE recall only drops a little when the sentence classifier is used, despite the fact that its recall on relevant sentences is only 63% in terrorism and 72% for disease outbreaks. One possible explanation is that the answer keys often contain multiple acceptable answer strings (e.g., “John Kennedy” and “JFK” might both be acceptable answers). On average, the answer keys contain approximately 1.64 acceptable strings per answer in the terrorism domain, and 1.77 acceptable strings per answer in the disease outbreaks do-

[11] This happens due to coreference, e.g., when multiple occurrences of an answer appear in a document, some of them may occur in relevant sentences while others do not.

Terrorism							
Patterns	*App*	*Rec*	*Pr*	*F*	*Rec*	*Pr*	*F*
		PerpInd			**PerpOrg**		
ASlogTS	All	.49	.35	.41	.33	.49	.40
ASlogTS	Rel	.41	.50	.45	.27	.58	.37
		Target			**Victim**		
ASlogTS	All	.64	.42	.51	.52	.48	.50
ASlogTS	Rel	.57	.49	.53	.48	.54	.51
		Weapon					
ASlogTS	All	.45	.39	.42			
ASlogTS	Rel	.40	.51	.45			
Disease Outbreaks							
		Disease			**Victim**		
ASlogTS	All	.51	.27	.36	.48	.35	.41
ASlogTS	Rel	.46	.31	.37	.44	.38	.41

Table 2: AutoSlog-TS Results

		Disease			**Victim**		
Patterns	*App*	*Rec*	*Pr*	*F*	*Rec*	*Pr*	*F*
ASlogTS	All	.51	.27	.36	.48	.35	.41
SA-50	All	.51	.25	.34	.47	.41	**.44**
SA-50	Rel	.49	.31	**.38**	.44	.43	.43
SA-50	Sel	.50	.29	.36	.46	.41	**.44**
SA-100	All	.57	.22	.32	.52	.33	.40
SA-100	Rel	.55	.28	.37	.49	.36	.41
SA-100	Sel	.56	.26	.35	.51	.34	.41
SA-150	All	.66	.20	.31	.55	.27	.37
SA-150	Rel	.61	.26	.36	.51	.31	.38
SA-150	Sel	.63	.24	.35	.53	.29	.37
SA-200	All	.68	.19	.30	.56	.26	.36
SA-200	Rel	.63	.25	.35	.52	.30	.38
SA-200	Sel	.65	.23	.34	.54	.28	.37

Table 3: ProMed Disease Outbreak Results

main. Thus, even if the sentence classifier discards some relevant sentences, an equally acceptable answer may be found in a different sentence.

6.3 Information Extraction Results

We first conducted two experiments with an existing IE pattern learner, AutoSlog-TS (Riloff, 1996) to give us a baseline against which to compare our results. The "All" rows in Table 2 show these results, where "All" means that the IE patterns were applied to all of the sentences in the test set. AutoSlog-TS[12] produced F scores between 40-51% on the MUC-4 test set, and 36-41% on the ProMed test set. The terrorism scores are competitive with the MUC-4 scores reported by Chieu et al. (2003), although they are not directly comparable because those scores are based on template generation. Since we created the ProMed test set ourselves, we are the first to report results on it[13].

Next, we evaluated the performance of AutoSlog-TS' extraction patterns when they are applied only in the sentences deemed to be relevant by our relevant sentence classifier. The purpose of this experiment was to determine whether the relevant sentence classifier can be beneficial when used with IE patterns known to be of good quality. The "Rel" rows in Table 2 show the scores for this experiment. Precision increased substantially on all 7 roles, although with some recall loss. This shows that a sentence classifier that has a high precision on irrelevant sentences but only a moderate precision on relevant sentences can be useful for information extraction.

Tables 3 and 4 show the results of our IE system, which uses the top N Semantic Affinity (SA) patterns and the relevant sentence classifier. We also show the AutoSlog-TS results again in the top row for comparison. The best F score for each role is shown in boldface. We used a lower probability threshold θ_l of 0.5 to filter out irrelevant patterns. We then ranked the remaining patterns based on semantic affinity, and evaluated the performance of the top 50, 100, 150, and 200 patterns. The *App* column indicates how the patterns were applied: for *All* they were applied in all sentences in the test set, for *Rel* they were applied only in the relevant sentences (as judged by our sentence classifier). For the *Sel* condition, the Primary patterns were applied in all sentences but the Secondary patterns were applied only in relevant sentences. To separate Primary and Secondary patterns we used an upper probability threshold θ_u of 0.8.

Looking at the rows with the *All* condition, we see that the semantic affinity patterns achieve good recall (e.g., the top 200 patterns have a recall over 50% for most roles), but precision is often quite low. This is not surprising because high semantic affinity patterns do not necessarily have to be relevant to the domain, so long as they recognize semantically appropriate things.

[12] AutoSlog-TS was trained on a much larger data set of 4,958 ProMed and 10,191 PubMed documents for the disease outbreaks domain. AutoSlog-TS requires a human review of the top-ranked patterns, which resulted in 396 patterns for the terrorism domain and 125 patterns for the disease outbreaks domain.

[13] Some previous work has been done with ProMed articles (Grishman et al., 2002a; Grishman et al., 2002b), but we are not aware of any IE evaluations on them.

		PerpInd			**PerpOrg**			**Target**			**Victim**			**Weapon**		
Patterns	*App*	*Rec*	*Pr*	*F*	*Rec*	*Pr*	*F*	*Rec*	*Pr*	*F*	*Rec*	*Pr*	*F*	*Rec*	*Pr*	*F*
ASlogTS	All	.49	.35	.41	.33	.49	.40	.64	.42	**.51**	.52	.48	**.50**	.45	.39	.42
SA-50	All	.24	.29	.26	.20	.42	.27	.42	.43	.42	.41	.43	.42	.53	.46	**.50**
SA-50	Rel	.19	.32	.24	.18	.60	.28	.38	.48	.42	.37	.52	.43	.41	.56	.48
SA-50	Sel	.20	.33	.25	.20	.54	.29	.42	.50	.45	.38	.52	.44	.43	.53	.48
SA-100	All	.40	.30	.34	.30	.43	.35	.56	.38	.45	.45	.37	.41	.55	.43	.48
SA-100	Rel	.36	.39	.38	.25	.59	.35	.52	.45	.48	.40	.47	.44	.45	.51	.48
SA-100	Sel	.38	.40	.39	.27	.55	.36	.56	.46	.50	.41	.47	.44	.47	.49	.48
SA-150	All	.50	.27	.35	.34	.39	.37	.62	.30	.40	.50	.33	.40	.55	.39	.45
SA-150	Rel	.46	.39	.42	.28	.58	.38	.56	.37	.45	.44	.45	.45	.45	.50	.47
SA-150	Sel	.48	.39	**.43**	.31	.55	.40	.60	.37	.46	.46	.44	.45	.47	.47	.47
SA-200	All	.73	.08	.15	.42	.43	.42	.64	.29	.40	.54	.32	.40	.64	.17	.27
SA-200	Rel	.67	.15	.24	.34	.61	.43	.58	.36	.45	.47	.43	.45	.52	.29	.37
SA-200	Sel	.71	.12	.21	.36	.58	**.45**	.61	.35	.45	.48	.43	.45	.53	.22	.31

Table 4: MUC-4 Terrorism Results

Next, we can compare each *All* row with the *Rel* row immediately below it. We observe that in every case precision improves, often dramatically. This demonstrates that our sentence classifier is having the desired effect. However, observe that the precision gain comes with some loss in recall points.

Clearly, this drop in recall is due to the answers embedded inside relevant sentences incorrectly classified as irrelevant. To counter this, we apply the Primary patterns to all the sentences. Thus, if we compare each *Rel* row with the *Sel* row immediately below it, we see the effect of loosening the reins on the Primary patterns (the Secondary patterns are still restricted to the relevant sentences). In most cases, the recall improves with a relatively small drop in precision, or no drop at all. In the terrorism domain, the highest F score for four of the five roles occurs under the *Sel* condition. In the disease outbreaks domain, the best F score for diseases occurs in the *Rel* condition, while the best score for victims is achieved under both the *All* and the *Sel* conditions.

Finally, we note that the best F scores produced by our information extraction system are higher than those produced by AutoSlog-TS for all of the roles except Targets and Victims, and our best performance on Targets is only slightly lower. These results are particularly noteworthy because AutoSlog-TS requires a human to manually review the patterns and assign event roles to them. In contrast, our approach is fully automated.

These results validate our hypothesis that decoupling the processes of finding relevant regions and applying semantically appropriate patterns can create an effective IE system.

7 Conclusions

In this work, we described an information extraction system based on a relevant sentence classifier and extraction patterns learned using a *semantic affinity* metric. The sentence classifier was self-trained using only relevant and irrelevant documents plus a handful of seed extraction patterns. We showed that separating the task of relevant region identification from that of pattern extraction can be effective for information extraction. In addition, we observed that the use of a relevant sentence classifier is beneficial for an IE system.

There are several avenues that need to be explored for future work. First, it would be interesting to see if the use of richer features can improve classifier performance, and if that in turn improves the performance of the IE system. We would also like to experiment with different region sizes and study their effect on information extraction. Finally, other techniques for learning semantically appropriate extraction patterns could be investigated.

Acknowledgments

This research was supported by NSF Grant IIS-0208985, Department of Homeland Security Grant N0014-07-1-0152, and the Institute for Scientific Computing Research and the Center for Applied Scientific Computing within Lawrence Livermore National Laboratory. We are grateful to Sean Igo and Rich Warren for annotating the disease outbreaks corpus.

References

E. Agichtein and L. Gravano. 2000. Snowball: Extracting Relations from Large Plain-Text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, San Antonio, TX, June.

R. Bunescu and R. Mooney. 2004. Collective Information Extraction with Relational Markov Networks. In *Proceeding of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 438–445, Barcelona, Spain, July.

M. Califf and R. Mooney. 1999. Relational Learning of Pattern-matching Rules for Information Extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, Orlando, FL, July.

J. Callan. 1994. Passage-Level Evidence in Document Retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310, Dublin, Ireland, July.

H. Chieu, H. Ng, and Y. Lee. 2003. Closing the Gap: Learning-Based Information Extraction Rivaling Knowledge-Engineering Methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 216–223, Sapporo, Japan, July.

F. Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalisation. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence*, pages 1251–1256, Seattle, WA, August.

C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

D. Freitag and A. McCallum. 2000. Information Extraction with HMM Structures Learned by Stochastic Optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 584–589, Austin, TX, August.

D. Freitag. 1998. Toward General-Purpose Learning for Information Extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 404–408, Montreal, Quebec, August.

R. Grishman, S. Huttunen, and R. Yangarber. 2002a. Information Extraction for Enhanced Access to Disease Outbreak Reports. *Journal of Biomedical Informatics*, 35(4):236–246, August.

R. Grishman, S. Huttunen, and R. Yangarber. 2002b. Real-Time Event Extraction for Infectious Disease Outbreaks. In *Proceedings of the 3rd Annual Human Language Technology Conference*, San Diego, CA, March.

J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. 1997. FASTUS: A Cascaded Finite-state Transducer for Extracting Information for Natural-Language Text. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*, pages 383–406. MIT Press, Cambridge, MA.

S. Huffman. 1996. Learning Information Extraction Patterns from Examples. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 246–260. Springer, Berlin.

T. Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the Tenth European Conference on Machine Learning*, pages 137–142, April.

J. Kim and D. Moldovan. 1993. PALKA: A System for Lexical Knowledge Acquisition. In *Proceedings of the Second International Conference on Information and Knowledge Management*, pages 124–131, Washington, DC, November.

S. Patwardhan and E. Riloff. 2006. Learning Domain-Specific Information Extraction Patterns from the Web. In *Proceedings of the ACL 2006 Workshop on Information Extraction Beyond the Document*, pages 66–73, Sydney, Australia, July.

A. Popescu, A. Yates, and O. Etzioni. 2004. Class Extraction from the World Wide Web. In Ion Muslea, editor, *Adaptive Text Extraction and Mining: Papers from the 2004 AAAI Workshop*, pages 68–73, San Jose, CA, July.

E. Riloff and W. Phillips. 2004. An Introduction to the Sundance and AutoSlog Systems. Technical Report UUCS-04-015, School of Computing, University of Utah.

E. Riloff. 1993. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816, Washington, DC, July.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Articial Intelligence*, pages 1044–1049, Portland, OR, August.

G. Salton, J. Allan, and C. Buckley. 1993. Approaches to Passage Retrieval in Full Text Information Systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 49–58, Pittsburgh, PA, June.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. 1995. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319, Montreal, Canada, August.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272, February.

K. Sudo, S. Sekine, and R. Grishman. 2003. An Improved Extraction Patterns Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 224–231, Sapporo, Japan, July.

B. Sundheim. 1992. Overview of the Fourth Message Understanding Evaluation and Conference. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 3–21, McLean, VA, June.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer, New York, NY.

A. Yakushiji, Y. Miyao, T. Ohta, Y. Tateisi, and J. Tsujii. 2006. Construction of Predicate-argument Structure Patterns for Biomedical Information Extraction. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 284–292, Sydney, Australia, July.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 940–946, Saarbrücken, Germany, August.

R. Yangarber, W. Lin, and R. Grishman. 2002. Unsupervised Learning of Generalized Names. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 154–160, Taipei, Taiwan, August.

Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information

GuoDong ZHOU[1][2] **Min ZHANG**[2] **Dong Hong JI**[2] **QiaoMing ZHU**[1]

[1]School of Computer Science & Technology
Soochow Univ.
Suzhou, China 215006
Email: {gdzhou,qmzhu}@suda.edu.cn

[2] Institute for Infocomm Research
Heng Mui Keng Terrace
Singapore 119613
Email: {zhougd, mzhang, dhji}@i2r.a-star.edu.sg

Abstract

This paper proposes a tree kernel with context-sensitive structured parse tree information for relation extraction. It resolves two critical problems in previous tree kernels for relation extraction in two ways. First, it automatically determines a dynamic context-sensitive tree span for relation extraction by extending the widely-used Shortest Path-enclosed Tree (SPT) to include necessary context information outside SPT. Second, it proposes a context-sensitive convolution tree kernel, which enumerates both context-free and context-sensitive sub-trees by considering their ancestor node paths as their contexts. Moreover, this paper evaluates the complementary nature between our tree kernel and a state-of-the-art linear kernel. Evaluation on the ACE RDC corpora shows that our dynamic context-sensitive tree span is much more suitable for relation extraction than SPT and our tree kernel outperforms the state-of-the-art Collins and Duffy's convolution tree kernel. It also shows that our tree kernel achieves much better performance than the state-of-the-art linear kernels. Finally, it shows that feature-based and tree kernel-based methods much complement each other and the composite kernel can well integrate both flat and structured features.

1 Introduction

Relation extraction is to find various predefined semantic relations between pairs of entities in text. The research in relation extraction has been promoted by the Message Understanding Conferences (MUCs) (MUC, 1987-1998) and the NIST Automatic Content Extraction (ACE) program (ACE, 2002-2005). According to the ACE Program, an entity is an object or a set of objects in the world and a relation is an explicitly or implicitly stated relationship among entities. For example, the sentence "Bill Gates is the chairman and chief software architect of Microsoft Corporation." conveys the ACE-style relation " EMPLOYMENT.exec" between the entities "Bill Gates" (person name) and "Microsoft Corporation" (organization name). Extraction of semantic relations between entities can be very useful in many applications such as question answering, e.g. to answer the query "Who is the president of the United States?", and information retrieval, e.g. to expand the query " George W. Bush" with "the president of the United States" via his relationship with "the United States".

Many researches have been done in relation extraction. Among them, feature-based methods (Kambhatla 2004; Zhou et al., 2005) achieve certain success by employing a large amount of diverse linguistic features, varying from lexical knowledge, entity-related information to syntactic parse trees, dependency trees and semantic information. However, it is difficult for them to effectively capture structured parse tree information (Zhou et al 2005), which is critical for further performance improvement in relation extraction.

As an alternative to feature-based methods, tree kernel-based methods provide an elegant solution to explore implicitly structured features by directly computing the similarity between two trees. Although earlier researches (Zelenko et al 2003; Culotta and Sorensen 2004; Bunescu and Mooney 2005a) only achieve success on simple tasks and fail on complex tasks, such as the ACE RDC task, tree kernel-based methods achieve much progress recently. As the state-of-the-art, Zhang et al (2006) applied the convolution tree kernel (Collins and Duffy 2001) and achieved comparable performance with a state-of-the-art linear kernel (Zhou et al 2005) on the 5 relation types in the ACE RDC 2003 corpus.

However, there are two problems in Collins and Duffy's convolution tree kernel for relation extraction. The first is that the sub-trees enumerated in the tree kernel computation are context-free. That is, each sub-tree enumerated in the tree kernel computation

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 728–736, Prague, June 2007. ©2007 Association for Computational Linguistics

does not consider the context information outside the sub-tree. The second is to decide a proper tree span in relation extraction. Zhang et al (2006) explored five tree spans in relation extraction and it was a bit surprising to find that the Shortest Path-enclosed Tree (SPT, i.e. the sub-tree enclosed by the shortest path linking two involved entities in the parse tree) performed best. This is contrast to our intuition. For example, "got married" is critical to determine the relationship between "John" and "Mary" in the sentence "John and Mary got married…" as shown in Figure 1(e). It is obvious that the information contained in SPT ("John and Marry") is not enough to determine their relationship.

This paper proposes a context-sensitive convolution tree kernel for relation extraction to resolve the above two problems. It first automatically determines a dynamic context-sensitive tree span for relation extraction by extending the Shortest Path-enclosed Tree (SPT) to include necessary context information outside SPT. Then it proposes a context-sensitive convolution tree kernel, which not only enumerates context-free sub-trees but also context-sensitive sub-trees by considering their ancestor node paths as their contexts. Moreover, this paper evaluates the complementary nature of different linear kernels and tree kernels via a composite kernel.

The layout of this paper is as follows. In Section 2, we review related work in more details. Then, the dynamic context-sensitive tree span and the context-sensitive convolution tree kernel are proposed in Section 3 while Section 4 shows the experimental results. Finally, we conclude our work in Section 5.

2 Related Work

The relation extraction task was first introduced as part of the Template Element task in MUC6 and then formulated as the Template Relation task in MUC7. Since then, many methods, such as feature-based (Kambhatla 2004; Zhou et al 2005, 2006), tree kernel-based (Zelenko et al 2003; Culotta and Sorensen 2004; Bunescu and Mooney 2005a; Zhang et al 2006) and composite kernel-based (Zhao and Grishman 2005; Zhang et al 2006), have been proposed in literature.

For the feature-based methods, Kambhatla (2004) employed Maximum Entropy models to combine diverse lexical, syntactic and semantic features in relation extraction, and achieved the F-measure of 52.8 on the 24 relation subtypes in the ACE RDC 2003 corpus. Zhou et al (2005) further systematically explored diverse features through a linear kernel and Support Vector Machines, and achieved the F-measures of 68.0 and 55.5 on the 5 relation types and the 24 relation subtypes in the ACE RDC 2003 corpus respectively. One problem with the feature-based methods is that they need extensive feature engineering. Another problem is that, although they can explore some structured information in the parse tree (e.g. Kambhatla (2004) used the non-terminal path connecting the given two entities in a parse tree while Zhou et al. (2005) introduced additional chunking features to enhance the performance), it is found difficult to well preserve structured information in the parse trees using the feature-based methods. Zhou et al (2006) further improved the performance by exploring the commonality among related classes in a class hierarchy using hierarchical learning strategy.

As an alternative to the feature-based methods, the kernel-based methods (Haussler, 1999) have been proposed to implicitly explore various features in a high dimensional space by employing a kernel to calculate the similarity between two objects directly. In particular, the kernel-based methods could be very effective at reducing the burden of feature engineering for structured objects in NLP researches, e.g. the tree structure in relation extraction.

Zelenko et al. (2003) proposed a kernel between two parse trees, which recursively matches nodes from roots to leaves in a top-down manner. For each pair of matched nodes, a subsequence kernel on their child nodes is invoked. They achieved quite success on two simple relation extraction tasks. Culotta and Sorensen (2004) extended this work to estimate similarity between augmented dependency trees and achieved the F-measure of 45.8 on the 5 relation types in the ACE RDC 2003 corpus. One problem with the above two tree kernels is that matched nodes must be at the same height and have the same path to the root node. Bunescu and Mooney (2005a) proposed a shortest path dependency tree kernel, which just sums up the number of common word classes at each position in the two paths, and achieved the F-measure of 52.5 on the 5 relation types in the ACE RDC 2003 corpus. They argued that the information to model a relationship between two entities can be typically captured by the shortest path between them in the dependency graph. While the shortest path may not be able to well preserve structured dependency tree information, another problem with their kernel is that the two paths should have same length. This makes it suffer from the similar behavior with that of Culotta and Sorensen (2004): high precision but very low recall.

As the state-of-the-art tree kernel-based method, Zhang et al (2006) explored various structured feature spaces and used the convolution tree kernel over parse trees (Collins and Duffy 2001) to model syntactic structured information for relation extraction. They achieved the F-measures of 61.9 and 63.6 on the 5 relation types of the ACE RDC 2003 corpus and the 7 relation types of the ACE RDC 2004 corpus respectively without entity-related information while the F-measure on the 5 relation types in the ACE RDC 2003 corpus reached 68.7 when entity-related information was included in the parse tree. One problem with Collins and Duffy's convolution tree kernel is that the sub-trees involved in the tree kernel computation are context-free, that is, they do not consider the information outside the sub-trees. This is different from the tree kernel in Culota and Sorensen (2004), where the sub-trees involved in the tree kernel computation are context-sensitive (that is, with the path from the tree root node to the sub-tree root node in consideration). Zhang et al (2006) also showed that the widely-used Shortest Path-enclosed Tree (SPT) performed best. One problem with SPT is that it fails to capture the contextual information outside the shortest path, which is important for relation extraction in many cases. Our random selection of 100 positive training instances from the ACE RDC 2003 training corpus shows that ~25% of the cases need contextual information outside the shortest path. Among other kernels, Bunescu and Mooney (2005b) proposed a subsequence kernel and applied it in protein interaction and ACE relation extraction tasks.

In order to integrate the advantages of feature-based and tree kernel-based methods, some researchers have turned to composite kernel-based methods. Zhao and Grishman (2005) defined several feature-based composite kernels to integrate diverse features for relation extraction and achieved the F-measure of 70.4 on the 7 relation types of the ACE RDC 2004 corpus. Zhang et al (2006) proposed two composite kernels to integrate a linear kernel and Collins and Duffy's convolution tree kernel. It achieved the F-measure of 70.9/57.2 on the 5 relation types/24 relation subtypes in the ACE RDC 2003 corpus and the F-measure of 72.1/63.6 on the 7 relation types/23 relation subtypes in the ACE RDC 2004 corpus.

The above discussion suggests that structured information in the parse tree may not be fully utilized in the previous works, regardless of feature-based, tree kernel-based or composite kernel-based methods. Compared with the previous works, this paper proposes a dynamic context-sensitive tree span trying to cover necessary structured information and a context-sensitive convolution tree kernel considering both context-free and context-sensitive sub-trees. Furthermore, a composite kernel is applied to combine our tree kernel and a state-of-the-art linear kernel for integrating both flat and structured features in relation extraction as well as validating their complementary nature.

3 Context Sensitive Convolution Tree Kernel for Relation Extraction

In this section, we first propose an algorithm to dynamically determine a proper context-sensitive tree span and then a context-sensitive convolution tree kernel for relation extraction.

3.1 Dynamic Context-Sensitive Tree Span in Relation Extraction

A relation instance between two entities is encapsulated by a parse tree. Thus, it is critical to understand which portion of a parse tree is important in the tree kernel calculation. Zhang et al (2006) systematically explored seven different tree spans, including the Shortest Path-enclosed Tree (SPT) and a Context-Sensitive Path-enclosed Tree[1] (CSPT), and found that SPT performed best. That is, SPT even outperforms CSPT. This is contrary to our intuition. For example, " got married" is critical to determine the relationship between "John" and " Mary" in the sentence "John and Mary got married…" as shown in Figure 1(e), and the information contained in SPT ("John and Mary") is not enough to determine their relationship. Obviously, context-sensitive tree spans should have the potential for better performance. One problem with the context-sensitive tree span explored in Zhang et al (2006) is that it only considers the availability of entities' siblings and fails to consider following two factors:

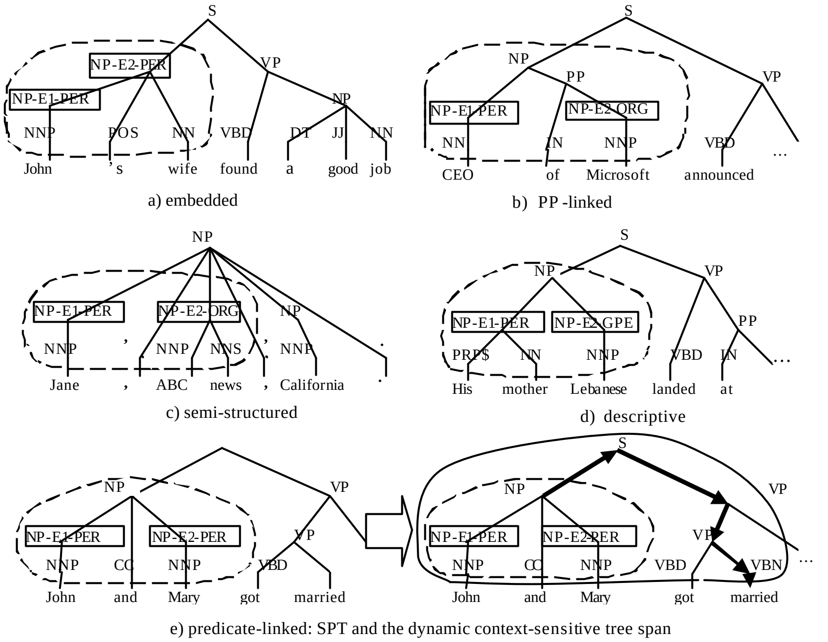
1) Whether is the information contained in SPT enough to determine the relationship between two entities? It depends. In the embedded cases, SPT is enough. For example, "John's wife" is enough to determine the relationship between "John" and " John's wife" in the sentence "John's wife got a good job…" as shown in Figure 1(a). However, SPT is not enough in the coordinated cases, e.g. to determine the relationship between "John" and " Mary" in the sentence "John and Mary got married…" as shown in Figure 1(e).

[1] CSPT means SPT extending with the 1st left sibling of the node of entity 1 and the 1st right sibling of the node of entity 2. In the case of no available sibling, it moves to the parent of current node and repeat the same process until a sibling is available or the root is reached.

2) How can we extend SPT to include necessary context information if there is no enough information in SPT for relation extraction?

To answer the above two questions, we randomly chose 100 positive instances from the ACE RDC 2003 training data and studied their necessary tree spans. It was observed that we can classify them into 5 categories: 1) embedded (37 instances), where one entity is embedded in another entity, e.g. "John" and "John's wife" as shown in Figure 1(a); 2) PP-linked (21 instances), where one entity is linked to another entity via PP attachment, e.g. "CEO" and "Microsoft" in the sentence "CEO of Microsoft announced …" as shown in Figure 1(b); 3) semi-structured (15 instances), where the sentence consists of a sequence of noun phrases (including the two given entities), e.g. "Jane" and "ABC news" in the sentence "Jane, ABC news, California." as shown in Figure 1(c); 4) descriptive (7 instances), e.g. the citizenship between " his mother" and "Lebanese" in the sentence "his mother Lebanese landed at …" as shown in Figure 1(d); 5) predicate-linked and others (19 instances, including coordinated cases), where the predicate information is necessary to determine the relationship between two entities, e.g. "John" and " Mary" in the sentence "John and Mary got married…" as shown in Figure 1(e);

Based on the above observations, we implement an algorithm to determine the necessary tree span for the relation extract task. The idea behind the algorithm is that the necessary tree span for a relation should be determined dynamically according to its tree span category and context. Given a parsed tree and two entities in consideration, it first determines the tree span category and then extends the tree span accordingly. By default, we adopt the Shortest Path-enclosed Tree (SPT) as our tree span. We only expand the tree span when the tree span belongs to the "predicate-linked" category. This is based on our observation that the tree spans belonging to the "predicate-linked" category vary much syntactically and majority (~70%) of them need information outside SPT while it is quite safe (>90%) to use SPT as the tree span for the remaining categories. In our algorithm, the expansion is done by first moving up until a predicate-headed phrase is found and then moving down along the predicated-headed path to the predicate terminal node. Figure 1(e) shows an example for the "predicate-linked" category where the lines with arrows indicate the expansion path.



a) embedded

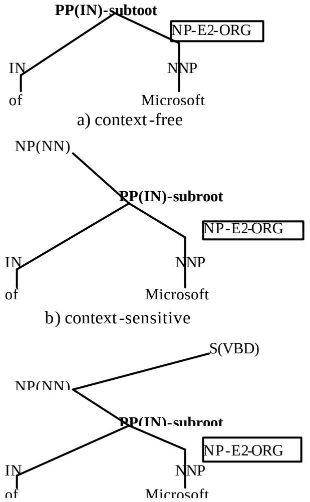
b) PP-linked

c) semi-structured

d) descriptive

e) predicate-linked: SPT and the dynamic context-sensitive tree span

Figure 1: Different tree span categories with SPT (dotted circle) and an example of the dynamic context-sensitive tree span (solid circle)



a) context-free

b) context-sensitive

c) context-sensitive

Figure 2: Examples of context-free and context-sensitive subtrees related with Figure 1(b). Note: the bold node is the root for a sub-tree.

A problem with our algorithm is how to determine whether an entity pair belongs to the "predicate-linked" category. In this paper, a simple method is applied by regarding the "predicate-linked" category as the default category. That is, those entity pairs, which do not belong to the four well defined and easily detected categories (i.e. embedded, PP-liked, semi-structured and descriptive), are classified into the "predicate-linked" category.

Since "predicate-linked" instances only occupy ~20% of cases, this explains why SPT performs better than the Context-Sensitive Path-enclosed Tree (CSPT) as described in Zhang et al (2006): consistently adopting CSPT may introduce too much noise/unnecessary information in the tree kernel.

3.2 Context-Sensitive Convolution Tree Kernel

Given any tree span, e.g. the dynamic context-sensitive tree span in the last subsection, we now study how to measure the similarity between two trees, using a convolution tree kernel. A convolution kernel (Haussler D., 1999) aims to capture structured information in terms of substructures. As a specialized convolution kernel, Collins and Duffy's convolution tree kernel $K_C(T_1,T_2)$ ('C' for convolution) counts the number of common sub-trees (sub-structures) as the syntactic structure similarity between two parse trees T_1 and T_2 (Collins and Duffy 2001):

$$K_C(T_1,T_2) = \sum_{n_1\in N_1, n_2\in N_2} \Delta(n_1,n_2) \qquad (1)$$

where N_j is the set of nodes in tree T_j , and $\Delta(n_1,n_2)$ evaluates the common sub-trees rooted at n_1 and n_2[2] and is computed recursively as follows:

1) If the context-free productions (Context-Free Grammar(CFG) rules) at n_1 and n_2 are different, $\Delta(n_1,n_2)=0$; Otherwise go to 2.
2) If both n_1 and n_2 are POS tags, $\Delta(n_1,n_2)=1\times l$; Otherwise go to 3.
3) Calculate $\Delta(n_1,n_2)$ recursively as:

$$\Delta(n_1,n_2) = l \prod_{k=1}^{\#ch(n_1)} (1+\Delta(ch(n_1,k),ch(n_2,k)) \qquad (2)$$

where $\#ch(n)$ is the number of children of node n, $ch(n,k)$ is the k^{th} child of node n and l (0< l <1) is the decay factor in order to make the kernel value less variable with respect to different sub-tree sizes.

This convolution tree kernel has been successfully applied by Zhang et al (2006) in relation extraction. However, there is one problem with this tree kernel: the sub-trees involved in the tree kernel computation are context-free (That is, they do not consider the information outside the sub-trees). This is contrast to the tree kernel proposed in Culota and Sorensen (2004) which is context-sensitive, that is, it considers the path from the tree root node to the sub-tree root node. In order to integrate the advantages of both tree kernels and resolve the problem in Collins and Duffy's convolution tree kernel, this paper proposes a context-sensitive convolution tree kernel. It works by taking ancestral information (i.e. the root node path) of sub-trees into consideration:

$$K_C(T[1],T[2]) = \sum_{i=1}^{m} \sum_{n_1^i[1]\in N_1^i[1], n_1^i[2]\in N_1^i[2]} \Delta(n_1^i[1], n_1^i[2]) \quad (3)$$

Where

- $N_1^i[j]$ is the set of root node paths with length *i* in tree *T[j]* while the maximal length of a root node path is defined by *m*.
- $n_1^i[j]=(n_1 n_2 \ldots n_i)[j]$ is a root node path with length *i* in tree *T[j]*, which takes into account the i-1 ancestral nodes n_2^i *[j]* of n_1*[j]* in *T[j]*. Here, $n_{k+1}[j]$ is the parent of $n_k[j]$ and $n_1[j]$ is the root node of a context-free sub-tree in *T[j]*. For better differentiation, the label of each ancestral node in n_1^i *[j]* is augmented with the POS tag of its head word.
- $\Delta(n_1^i[1], n_1^i[2])$ measures the common context-sensitive sub-trees rooted at root node paths $n_1^i[1]$ and $n_1^i[2]$[3]. In our tree kernel, a sub-tree becomes context-sensitive with its dependence on the root node path instead of the root node itself. Figure 2 shows a few examples of context-sensitive sub-trees with comparison to context-free sub-trees.

Similar to Collins and Duffy (2001), our tree kernel computes $\Delta(n_1^i[1], n_1^i[2])$ recursively as follows:

1) If the context-sensitive productions (Context-Sensitive Grammar (CSG) rules with root node paths as their left hand sides) rooted at $n_1^i[1]$ and $n_1^i[2]$ are different, return $\Delta(n_1^i[1], n_1^i[2])$ =0; Otherwise go to Step 2.
2) If both $n_1[1]$ and $n_1[2]$ are POS tags, $\Delta(n_1^i[1], n_1^i[2]) = l$; Otherwise go to Step 3.

[2] That is, each node n encodes the identity of a sub-tree rooted at n and, if there are two nodes in the tree with the same label, the summation will go over both of them.

[3] That is, each root node path n_1^i encodes the identity of a context-sensitive sub-tree rooted at n_1^i and, if there are two root node paths in the tree with the same label sequence, the summation will go over both of them.

3) Calculate $\Delta(n_1^i[1], n_1^i[2])$ recursively as:

$$\Delta(n_1^i[1], n_1^i[2]) = l \prod_{k=1}^{\#ch(n_1^i[1])} (1 + \Delta(ch(n_1^i[1], k), ch(n_1^i[2], k)) \quad (4)$$

where $ch(n_1^i[j], k])$ is the k^{th} context-sensitive child of the context-sensitive sub-tree rooted at $n_1^i[j]$ with $\#ch(n_1^i[j])$ the number of the context-sensitive children. Here, l (0< l <1) is the decay factor in order to make the kernel value less variable with respect to different sizes of the context-sensitive sub-trees.

It is worth comparing our tree kernel with previous tree kernels. Obviously, our tree kernel is an extension of Collins and Duffy's convolution tree kernel, which is a special case of our tree kernel (if *m=1* in Equation (3)). Our tree kernel not only counts the occurrence of each context-free sub-tree, which does not consider its ancestors, but also counts the occurrence of each context-sensitive sub-tree, which considers its ancestors. As a result, our tree kernel is not limited by the constraints in previous tree kernels (as discussed in Section 2), such as Collins and Duffy (2001), Zhang et al (2006), Culotta and Sorensen (2004) and Bunescu and Mooney (2005a). Finally, let's study the computational issue with our tree kernel. Although our tree kernel takes the context-sensitive sub-trees into consideration, it only slightly increases the computational burden, compared with Collins and Duffy's convolution tree kernel. This is due to that $\Delta(n_1[1], n_1[2]) = 0$ holds for the majority of context-free sub-tree pairs (Collins and Duffy 2001) and that computation for context-sensitive sub-tree pairs is necessary only when $\Delta(n_1[1], n_1[2]) \neq 0$ and the context-sensitive sub-tree pairs have the same root node path(i.e. $n_1^i[1] = n_1^i[2]$ in Equation (3)).

4 Experimentation

This paper uses the ACE RDC 2003 and 2004 corpora provided by LDC in all our experiments.

4.1 Experimental Setting

The ACE RDC corpora are gathered from various newspapers, newswire and broadcasts. In the 2003 corpus, the training set consists of 674 documents and 9683 positive relation instances while the test set consists of 97 documents and 1386 positive relation instances. The 2003 corpus defines **5** entity types, **5** major relation types and **24** relation subtypes. All the reported performances in this paper on the ACE RDC 2003 corpus are evaluated on the test data. The 2004 corpus contains 451 documents and 5702 positive relation instances. It redefines **7** entity types, **7** major relation types and **23** relation subtypes. For comparison, we use the same setting as Zhang et al (2006) by applying a 5-fold cross-validation on a subset of the 2004 data, containing 348 documents and 4400 relation instances. That is, all the reported performances in this paper on the ACE RDC 2004 corpus are evaluated using 5-fold cross validation on the entire corpus.

Both corpora are parsed using Charniak's parser (Charniak, 2001) with the boundaries of all the entity mentions kept[4]. We iterate over all pairs of entity mentions occurring in the same sentence to generate potential relation instances[5]. In our experimentation, SVM (SVMLight, Joachims(1998)) is selected as our classifier. For efficiency, we apply the *one vs. others* strategy, which builds K classifiers so as to separate one class from all others. The training parameters are chosen using cross-validation on the ACE RDC 2003 training data. In particular, l in our tree kernel is fine-tuned to 0.5. This suggests that about 50% discount is done as our tree kernel moves down one level in computing $\Delta(n_1^i[1], n_1^i[2])$.

4.2 Experimental Results

First, we systematically evaluate the context-sensitive convolution tree kernel and the dynamic context-sensitive tree span proposed in this paper.

Then, we evaluate the complementary nature between our tree kernel and a state-of-the-art linear kernel via a composite kernel. Generally different feature-based methods and tree kernel-based methods have their own merits. It is usually easy to build a system using a feature-based method and achieve the state-of-the-art performance, while tree kernel-based methods hold the potential for further performance improvement. Therefore, it is always a good idea to integrate them via a composite kernel.

[4] This can be done by first representing all entity mentions with their head words and then restoring all the entity mentions after parsing. Moreover, please note that the final performance of relation extraction may change much with different range of parsing errors. We will study this issue in the near future.

[5] In this paper, we only measure the performance of relation extraction on " true" mentions with "true" chaining of co-reference (i.e. as annotated by LDC annotators). Moreover, we only model explicit relations and explicitly model the argument order of the two mentions involved.

Finally, we compare our system with the state-of-the-art systems in the literature.

Context-Sensitive Convolution Tree Kernel

In this paper, the *m* parameter of our context-sensitive convolution tree kernel as shown in Equation (3) indicates the maximal length of root node paths and is optimized to 3 using 5-fold cross validation on the ACE RDC 2003 training data. Table 1 compares the impact of different *m* in context-sensitive convolution tree kernels using the Shortest Path-enclosed Tree (SPT) (as described in Zhang et al (2006)) on the major relation types of the ACE RDC 2003 and 2004 corpora, in details. It also shows that our tree kernel achieves best performance on the test data using SPT with $m = 3$, which outperforms the one with $m = 1$ by ~2.3 in F-measure. This suggests the parent and grandparent nodes of a sub-tree contains much information for relation extraction while considering more ancestral nodes may not help. This may be due to that, although our experimentation on the training data indicates that more than 80% (on average) of subtrees has a root node path longer than 3 (since most of the subtrees are deep from the root node and more than 90% of the parsed trees in the training data are deeper than 6 levels), including a root node path longer than 3 may be vulnerable to the full parsing errors and have negative impact. Table 1 also evaluates the impact of entity-related information in our tree kernel by attaching entity type information (e.g. "PER" in the entity node 1 of Figure 1(b)) into both entity nodes. It shows that such information can significantly improve the performance by ~6.0 in F-measure. In all the following experiments, we will apply our tree kernel with *m*=3 and entity-related information by default.

Table 2 compares the dynamic context-sensitive tree span with SPT using our tree kernel. It shows that the dynamic tree span can futher improve the performance by ~1.2 in F-measure[6]. This suggests the usefulness of extending the tree span beyond SPT for the "predicate-linked" tree span category. In the future work, we will further explore expanding the dynamic tree span beyond SPT for the remaining tree span categories.

m	P(%)	R(%)	F
1	72.3(72.7)	56.6(53.8)	63.5(61.8)
2	74.9(75.2)	57.9(54.7)	65.3(63.5)
3	75.7(76.1)	58.3(55.1)	65.9(64.0)
4	76.0(75.9)	58.3(55.3)	66.0(63.9)

a) without entity-related information

m	P(%)	R(%)	F
1	77.2(76.9)	63.5(60.8)	69.7(67.9)
2	79.1(78.6)	65.0(62.2)	71.3(69.4)
3	79.6(79.4)	65.6(62.5)	71.9(69.9)
4	79.4(79.1)	65.6(62.3)	71.8(69.7)

b) with entity-related information

Table 1: Evaluation of context-sensitive convolution tree kernels using SPT on the major relation types of the ACE RDC 2003 (inside the parentheses) and 2004 (outside the parentheses) corpora.

Tree Span	P(%)	R(%)	F
Shortest Path-enclosed Tree	79.6 (79.4)	65.6 (62.5)	71.9 (69.9)
Dynamic Context-Sensitive Tee	81.1 (80.1)	66.7 (63.8)	73.2 (71.0)

Table 2: Comparison of dynamic context-sensitive tree span with SPT using our context-sensitive convolution tree kernel on the major relation types of the ACE RDC 2003 (inside the parentheses) and 2004 (outside the parentheses) corpora. 18% of positive instances in the ACE RDC 2003 test data belong to the predicate-linked category.

Composite Kernel

In this paper, a composite kernel via polynomial interpolation, as described Zhang et al (2006), is applied to integrate the proposed context-sensitive convolution tree kernel with a state-of-the-art linear kernel (Zhou et al 2005)[7]:

$$K_1(\bullet,\bullet) = a \cdot K_L^P(\bullet,\bullet) + (1-a)\cdot K_C(\bullet,\bullet) \quad (5)$$

Here, $K_L(\bullet,\bullet)$ and $K_C(\bullet,\bullet)$ indicates the normalized linear kernel and context-sensitive convolution tree kernel respectively while $K^p(\cdot,\cdot)$ is the polynomial expansion of $K(\cdot,\cdot)$ with degree *d*=2, i.e. $K^p(\cdot,\cdot) = (K(\cdot,\cdot)+1)^2$ and a is the coefficient (a is set to 0.3 using cross-validation).

[6] Significance test shows that the dynamic tree span performs statistically significantly better than SPT with p-values smaller than 0.05.

[7] Here, we use the same set of flat features (i.e. word, entity type, mention level, overlap, base phrase chunking, dependency tree, parse tree and semantic information) as Zhou et al (2005).

Table 3 evaluates the performance of the composite kernel. It shows that the composite kernel much further improves the performance beyond that of either the state-of-the-art linear kernel or our tree kernel and achieves the F-measures of 74.1 and 75.8 on the major relation types of the ACE RDC 2003 and 2004 corpora respectively. This suggests that our tree kernel and the state-of-the-art linear kernel are quite complementary, and that our composite kernel can effectively integrate both flat and structured features.

System	P(%)	R(%)	F
Linear Kernel	78.2 (77.2)	63.4 (60.7)	70.1 (68.0)
Context-Sensitive Convolution Tree Kernel	81.1 (80.1)	66.7 (63.8)	73.2 (71.0)
Composite Kernel	82.2 (80.8)	70.2 (68.4)	75.8 (74.1)

Table 3: Performance of the composite kernel via polynomial interpolation on the major relation types of the ACE RDC 2003 (inside the parentheses) and 2004 (outside the parentheses) corpora

Comparison with Other Systems

ACE RDC 2003	P(%)	R(%)	F
Ours: composite kernel	80.8 (65.2)	68.4 (54.9)	74.1 (59.6)
Zhang et al (2006): composite kernel	77.3 (64.9)	65.6 (51.2)	70.9 (57.2)
Ours: context-sensitive convolution tree kernel	80.1 (63.4)	63.8 (51.9)	71.0 (57.1)
Zhang et al (2006): convolution tree kernel	76.1 (62.4)	62.6 (48.5)	68.7 (54.6)
Bunescu et al (2005): shortest path dependency kernel	65.5 (-)	43.8 (-)	52.5 (-)
Culotta et al (2004): dependency kernel	67.1 (-)	35.0 (-)	45.8 (-)
Zhou et al. (2005): feature-based	77.2 (63.1)	60.7 (49.5)	68.0 (55.5)
Kambhatla (2004): feature-based	- (63.5)	- (45.2)	- (52.8)

Table 4: Comparison of difference systems on the ACE RDC 2003 corpus over both 5 types (outside the parentheses) and 24 subtypes (inside the parentheses)

ACE RDC 2004	P(%)	R(%)	F
Ours: composite kernel	82.2 (70.3)	70.2 (62.2)	75.8 (66.0)
Zhang et al (2006): composite kernel	76.1 (68.6)	68.4 (59.3)	72.1 (63.6)
Zhao et al (2005):[8] composite kernel	69.2 (-)	70.5 (-)	70.4 (-)
Ours: context-sensitive convolution tree kernel	81.1 (68.8)	66.7 (60.3)	73.2 (64.3)
Zhang et al (2006): convolution tree kernel	72.5 (-)	56.7 (-)	63.6 (-)

Table 5: Comparison of difference systems on the ACE RDC 2004 corpus over both 7 types (outside the parentheses) and 23 subtypes (inside the parentheses)

Finally, Tables 4 and 5 compare our system with other state-of-the-art systems[9] on the ACE RDC 2003 and 2004 corpora, respectively. They show that our tree kernel-based system outperforms previous tree kernel-based systems. This is largely due to the context-sensitive nature of our tree kernel which resolves the limitations of the previous tree kernels. They also show that our tree kernel-based system outperforms the state-of-the-art feature-based system. This proves the great potential inherent in the parse tree structure for relation extraction and our tree kernel takes a big stride towards the right direction. Finally, they also show that our composite kernel-based system outperforms other composite kernel-based systems.

5 Conclusion

Structured parse tree information holds great potential for relation extraction. This paper proposes a context-sensitive convolution tree kernel to resolve two critical problems in previous tree kernels for relation extraction by first automatically determining a dynamic context-sensitive tree span and then applying a context-sensitive convolution tree kernel. Moreover, this paper evaluates the complementary nature between our tree kernel and a state-of-the-art linear kernel. Evaluation on the ACE RDC corpora shows that our dynamic context-sensitive tree span is much more suitable for relation extraction than the widely-used Shortest Path-enclosed Tree and our tree kernel outperforms the state-of-the-art Collins and Duffy's convolution tree kernel. It also shows that feature-based

[8] There might be some typing errors for the performance reported in Zhao and Grishman(2005) since P, R and F do not match.

[9] All the state-of-the-art systems apply the entity-related information. It is not supervising: our experiments show that using the entity-related information gives a large performance improvement.

and tree kernel-based methods well complement each other and the composite kernel can effectively integrate both flat and structured features.

To our knowledge, this is the first research to demonstrate that, without extensive feature engineering, an individual tree kernel can achieve much better performance than the state-of-the-art linear kernel in relation extraction. This shows the great potential of structured parse tree information for relation extraction and our tree kernel takes a big stride towards the right direction.

For the future work, we will focus on improving the context-sensitive convolution tree kernel by exploring more useful context information. Moreover, we will explore more entity-related information in the parse tree. Our preliminary work of including the entity type information significantly improves the performance. Finally, we will study how to resolve the data imbalance and sparseness issues from the learning algorithm viewpoint.

Acknowledgement

This research is supported by Project 60673041 under the National Natural Science Foundation of China and Project 2006AA01Z147 under the “863” National High-Tech Research and Development of China. We would also like to thank the critical and insightful comments from the four anonymous reviewers.

References

ACE. (2000-2005). Automatic Content Extraction. http://www.ldc.upenn.edu/Projects/ACE/

Bunescu R. & Mooney R.J. (2005a). A shortest path dependency kernel for relation extraction. *HLT/EMNLP'2005*: 724-731. 6-8 Oct 2005. Vancover, B.C.

Bunescu R. & Mooney R.J. (2005b). Subsequence Kernels for Relation Extraction *NIPS'2005*. Vancouver, BC, December 2005

Charniak E. (2001). Immediate-head Parsing for Language Models. *ACL'2001*: 129-137. Toulouse, France

Collins M. and Duffy N. (2001). Convolution Kernels for Natural Language. *NIPS'2001*: 625-632. Cambridge, MA

Culotta A. and Sorensen J. (2004). Dependency tree kernels for relation extraction. *ACL'2004*. 423-429. 21-26 July 2004. Barcelona, Spain.

Haussler D. (1999). Convolution Kernels on Discrete Structures. *Technical Report UCS-CRL-99-10*, University of California, Santa Cruz

Joachims T. (1998). Text Categorization with Support Vector Machine: learning with many relevant features. *ECML-1998*: 137-142. Chemnitz, Germany

Kambhatla N. (2004). Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. *ACL'2004*(Poster). 178-181. 21-26 July 2004. Barcelona, Spain.

MUC. (1987-1998). The NIST MUC website: http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

Zelenko D., Aone C. and Richardella. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*. 3(Feb):1083-1106.

Zhang M., Zhang J., Su J. and Zhou GD. (2006). A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. *COLING-ACL-2006*: 825-832. Sydney, Australia

Zhao S.B. and Grishman R. (2005). Extracting relations with integrated information using kernel methods. *ACL'2005*: 419-426. Univ of Michigan-Ann Arbor, USA, 25-30 June 2005.

Zhou G.D., Su J. Zhang J. and Zhang M. (2005). Exploring various knowledge in relation extraction. *ACL'2005*. 427-434. 25-30 June, Ann Arbor, Michgan, USA.

Zhou G.D., Su J. and Zhang M. (2006). Modeling commonality among related classes in relation extraction, *COLING-ACL'2006*: 121-128. Sydney, Australia.

Chinese Syntactic Reordering for Statistical Machine Translation

Chao Wang
MIT CSAIL
32 Vassar Street, Room 362
Cambridge, MA 02139, USA
wangc@csail.mit.edu

Michael Collins
MIT CSAIL
32 Vassar Street, Room G-484
Cambridge, MA 02139, USA
mcollins@csail.mit.edu

Philipp Koehn
School of Informatics
2 Buccleuch Place, 5BP 2L2
Edinburgh, EH8 9LW, UK
pkoehn@inf.ed.ac.uk

Abstract

Syntactic reordering approaches are an effective method for handling word-order differences between source and target languages in statistical machine translation (SMT) systems. This paper introduces a reordering approach for translation from Chinese to English. We describe a set of syntactic reordering rules that exploit systematic differences between Chinese and English word order. The resulting system is used as a preprocessor for both training and test sentences, transforming Chinese sentences to be much closer to English in terms of their word order. We evaluated the reordering approach within the MOSES phrase-based SMT system (Koehn et al., 2007). The reordering approach improved the BLEU score for the MOSES system from 28.52 to 30.86 on the NIST 2006 evaluation data. We also conducted a series of experiments to analyze the accuracy and impact of different types of reordering rules.

1 Introduction

Syntactic reordering approaches are an effective method for handling systematic differences in word order between source and target languages within the context of statistical machine translation (SMT) systems (Xia and McCord, 2004; Collins et al., 2005). In reordering approaches, sentences in the source language are first parsed, for example using a Treebank-trained parser. A series of transformations is then applied to the resulting parse tree, with the goal of transforming the source language sentence into a word order that is closer to that of the target language. The reordering process is used to preprocess both the training and test data used within an existing SMT system. Reordering approaches have given significant improvements in performance for translation from French to English (Xia and McCord, 2004) and from German to English (Collins et al., 2005).

This paper describes a syntactic reordering approach for translation from Chinese to English. Figure 1 gives an example illustrating some of the differences in word order between the two languages. The example shows a Chinese sentence whose literal translation in English is:

> *this is French delegation at Winter Olympics on achieve DEC best accomplishment*

and where a natural translation would be

> *this is the best accomplishment that the French delegation achieved at the Winter Olympics*

As exemplified by this sentence, Chinese differs from English in several important respects: for example, relative clauses appear *before* the noun being modified; prepositional phrases often appear *before* the head they modify; and so on. It can be seen that some significant reordering of the input is required to produce a good English translation. For this example, application of reordering rules leads to a new Chinese string whose word-by-word English paraphrase is:

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 737–745, Prague, June 2007. ©2007 Association for Computational Linguistics

Before syntactic reordering

```
IP NP PN 这(this)
   VP VC(is)
      NP CP IP NP NR 法国(French)
                  NN 代表团(delegation)
               VP PP P 在(at)
                     LCP NP NN 冬季
                               (Winter)
                            NR 奥运会
                               (Olympics)
                         LC 上(on)
                  VP-A VV 取得(achieve)
            DEC 的(DEC)
         ADJP JJ 最好(best)
         NPB NN 成绩(accomplishment)
```

After syntactic reordering

```
IP NP PN 这(this)
   VP VC(is)
      NP ADJP JJ 最好(best)
         NPB NN 成绩(accomplishment)
         CP DEC 的(DEC)
            IP NP NR 法国(French)
                  NN 代表团(delegation)
               VP VP-A VV 取得(achieve)
                  PP P 在(at)
                     LCP LC 上(on)
                         NP NN 冬季
                               (Winter)
                            NR 奥运会
                               (Olympics)
```

Figure 1: Original (left) and reordered (right) parse trees for the Chinese sentence “这是法国代表团在冬季奥运会上取得的最好成绩,” which translates into “*This is the best accomplishment that the French delegation achieved at the Winter Olympics*” in English.

> *this is best accomplishment DEC French delegation achieve at on Winter Olympics*

This reordering is relatively easy to express using syntactic transformations—for example, it is simple to move the entire relative clause “*French delegation at Winter Olympics on achieve DEC*” to a position that is after the noun phrase it modifies, namely “*best accomplishment*.” Phrase-based systems are quite limited in their ability to perform transformations of this type. More recently developed hierarchical systems (e.g., (Yamada and Knight, 2001; Chiang, 2005; Marcu et al., 2006)) may be better equipped to deal with reordering of this type; however, in this example they would effectively have to first identify the span of the relative clause, and then move it into the correct position, without any explicit representation of the source language syntax.

In this paper, we describe a set of syntactic reordering rules that exploit systematic differences between Chinese and English word order. The resulting system is used as a preprocessor for both training and test sentences, transforming Chinese sentences to be much closer to English. We report results for the method on the NIST 2006 evaluation data, using the MOSES phrase-based SMT system (Koehn et al., 2007). The reordering rules give an improvement in accuracy from 28.52 to 30.86 BLEU score. A concern for methods that make use of Chinese parsers is that these parsers are typically of relatively low accuracy, particularly given that Chinese requires a word-segmentation step that is not required in languages such as English. Our results show that Chinese parses are useful in SMT in spite of this problem. We report results showing the precision of the reordering rules—essentially testing how often the Chinese sentences are correctly reordered—to give more insight into this issue. We also report experiments which assess the impact of each type of reordering rule on translation accuracy.

2 Related Work

A number of researchers (Brown et al., 1992; Berger et al., 1996; Niessen and Ney, 2004; Xia and McCord, 2004; Collins et al., 2005) have described approaches that preprocess the source language input in SMT systems. We are not, however, aware of work on this topic for translation from Chinese to English. Brown et al. (1992) describe an analysis component for French which moves phrases around (in addition to other transformations) so the source and target sentences are closer to each other in word order. Berger et al. (1996) describe an approach for French that reorders phrases of the form $NOUN_1$ *de* $NOUN_2$. Xia and McCord (2004) describe an approach for French, where reordering rules that operate on context-free rule productions are acquired automatically. Niessen and Ney (2004) describe an approach for translation from German to English that combines verbs with associated particles, and also reorders questions. Collins et al. (2005) also describe an approach for German, concentrating on reordering German clauses, which have quite different word order from clauses in English. Our approach is most similar to that of Collins et al. (2005).

Most SMT systems employ some mechanism that allows reordering of the source language during translation (i.e., non-monotonic decoding). The MOSES phrase-based system that we use has a relatively simple reordering model which has a fixed penalty for reordering moves in the decoder. More sophisticated models include reordering parameters that are sensitive to lexical information (Tillmann, 2004; Kumar and Byrne, 2005; Koehn et al., 2005). The model of Chiang (2005) employs a synchronous context-free grammar to allow hierarchical approaches to reordering. The syntax-based models of Yamada and Knight (2001) and Marcu et al. (2006) build a full parse tree in the target language, again effectively allowing hierarchical reordering based on synchronous grammars. It is worth noting that none of these approaches to reordering make use of explicit syntactic information in the *source* language—for example, none of the methods make use of an existing source-language parser (the systems of Yamada and Knight (2001) and Marcu et al. (2006) make use of a parser in the target language, i.e., English).

Finally, note that a number of statistical MT systems make use of source language syntax in transducer-style approaches; see (Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006). In contrast to the preprocessing approach, they attempt to incorporate syntax directly into the decoding stage.

3 Chinese Syntactic Reordering Rules

We used the Penn Chinese Treebank guidelines (Xue et al., 2005) in searching for a suitable set of reordering rules. We examined all phrase types in the Treebank; potentially phrases of any type could be candidates for reordering rules. Table 1 provides a list of Treebank phrase tags for easy reference. We ruled out several phrase types as not requiring reordering rules. For example, Chinese ADJPs, ADVPs, DPs, QPs, and PPs all have similar internal word ordering to their English counterparts. Also similar are a group of special structures such as LST, FRAG, and PRN.

ADJP	adjective phrase
ADVP	adverbial phrase headed by AD (adverb)
CLP	classifier phrase
CP	clause headed by C (complementizer)
DNP	phrase formed by “XP+DEG”
DP	determiner phrase
DVP	phrase formed by “XP+DEV”
FRAG	fragment
IP	simple clause headed by I (INFL)
LCP	phrase formed by “XP+LC”
LST	list marker
NP	noun phrase
PP	preposition phrase
PRN	parenthetical
QP	quantifier phrase
UCP	unidentical coordination phrase
VP	verb phrase

Table 1: Penn Chinese Treebank phrase tags.

We identified three categories that we considered to be the most prominent candidates for reordering. These phrases include VPs (verb phrases), NPs (noun phrases), and LCPs (localizer phrases, which frequently map to prepositional phrases in English). In the following, we discuss each of the three main categories in more detail.

3.1 Verb Phrases

In Chinese, verb phrase modifiers typically occur in pre-verbal position. VP modifiers can be ADVPs, temporal and spatial NPs, QP, PPs, CPs, IPs, DVPs, and LCPs. The ADVPs are simple adverbs, which can occur both preverbal and postverbal in an English verb phrase, so we do not attempt to move them. Similarly, the CP, IP, and DVP modifiers are typically adverbial phrases, which do not have a fixed position in English verb phrases. In the following, we only consider cases involving PPs, LCPs, temporal and spatial NPs, and QPs.

PPs and LCPs Figure 2 shows an example verb phrase with a PP modifier, which translates literally

```
VP PP P 在(at)
      NP-A NPB NN 东部(Eastern)
                NN 联盟(Division)
   VP-A VV 名列(rank)
        QP OD 第十(10th)
```

Figure 2: Example VP with PP modifier. The phrase translates into "*ranks 10^{th} in the Eastern Division*."

```
VP NP NPB NT 当天(same day)
          NT 上午(morning)
   VP-A VV 发表(issue)
        NP-A NPB NN 声明(statement)
```

Figure 3: Example VP with temporal NP modifier. The phrase translates into "*issued a statement that morning*."

into "*at Eastern Division rank 10^{th}*." Recognizing that PPs in English verb phrases almost always occur after the verb, we use a simple VP(PP:VP) reordering rule which states that a PP in a parent VP needs to be repositioned after the sibling VP. LCPs are similar to PPs and typically map to prepositional phrases in English. Thus they are handled similarly to PPs, i.e., LCPs in a parent VP are repositioned after the sibling VP.

NPs Figure 3 gives an example of a verb phrase with a temporal NP modifier, which literally translates into "*same day morning issue statement*." In English, temporal phrases such as these almost always occur after the head verb. Conveniently, the Chinese Treebank uses the part of speech (POS) tag NT for temporal nouns. Thus, we use a rule which states that a preverbal NP will be repositioned after the sibling VP if there is at least one NT in the NP subtree. A similar rule might apply to locative NPS; however, there is no special POS tag in the Treebank marking locations,[1] so we do not have a syntax-based reordering rule to handle locative NPs.

QPs QP modifiers in verb phrases often correspond to time-related concepts such as duration and frequency. Figure 4 shows an example verb phrase with a QP modifier, literally translating into "*many time injured*." Since temporal phrases almost always occur after the verb in English verb phrases, we handle such cases by a simple rule which states that the QP in a parent VP will be repositioned after the sibling VP.

```
VP QP CD 多(many)
      CLP M 次(time)
   VP-A VV 受伤(injured)
```

Figure 4: Example VP with QP modifier. The phrase translates into "*injured many times*."

```
NP-A DNP PP P 对(to)
            NP-A NPB NR 津巴布韦(Zimbabwe)
         DEG 的(DEG)
     NPB NN 经济(financial)
         NN 援助(aid)
```

Figure 5: An example Chinese NP with a DNP modifier headed by a PP. The phrase translates into "*the financial aid to Zimbabwe*" in English.

3.2 Noun Phrases

Noun phrases in Chinese can take several types of modifiers: for example, phrases of type QP, DP, ADJP, NP, DNP, and CP. The placement of QP, DP, and ADJP modifiers is somewhat similar to English in that these phrases typically occur before the noun they modify. The case of NP modifiers in NPs is very limited in the Chinese Treebank, since most noun-noun sequences form compounds in a single NP. Hence we only developed reordering rules to handle DNP and clausal (CP) modifiers.

DNPs DNPs are formed by "XP+DEG," where XP can be a phrase of the type ADJP, QP, PP, LCP, or NP. When the XP is an ADJP or a QP, no reordering is needed because the word order is the same as that of English.

When the XP is a PP or an LCP, the DNP essentially corresponds to a prepositional phrase in English, which almost always appears after the noun being modified. Figure 5 shows an example where the XP in the DNP is a PP. The reordering rule to handle these two cases states that, if a parent NP has a child DNP which in turn has a child PP or LCP, then the DNP is repositioned after the last sibling NP.

Figure 6 shows an example noun phrase for which the XP in the DNP is NP. On the surface, the Chinese "NP_1 DEG NP_2" sequence is analogous to the English possessive structure of "NP_1's NP_2" and does

[1] One can argue that NR (proper nouns) in that context are likely to be places. However, there also exist many exceptions, and so we decided not to exploit the NR tag.

```
NP-A DNP NP DP DT 该(this)
            CLP M 项(measure word)
            NPB NN 技术(technique)
         DEG 的(DEG)
     NPB NN 掌握(mastery)
```

Figure 6: An example Chinese NP phrase with a DNP modifier headed by a NP. The phrase translates into "*the mastery of this technique*" in English.

not require reordering, for example, "苏(*Sue*) 的(*'s*) 朋友(*friend*)" in Chinese and "*Sue's friend*" in English. However, the Chinese possessive structure "NP$_1$ DEG NP$_2$" can express more sophisticated relationships which are inappropriate for the "NP$_1$'s NP$_2$" expression. For example, the phrase in Figure 6 can only be translated into "*the mastery of this technique*," but not "*this technique's mastery*." We decide to reorder DNPs of the "NP+DEG" format, because they often can only map to the "NP$_2$ of NP$_1$" expression in English. Additionally, the "NP$_2$ of NP$_1$" expression is more general and can replace "NP$_1$'s NP$_2$" in many cases. One exception is when the NP is a pronoun (PN), e.g., "他(*he*) 的(*'s*) 名字(*name*)," in which case the DNP acts simply like a possessive pronoun. Our reordering rule thus states that, if a parent NP has a child DNP which in turn has a child NP that is not a PN, then the DNP is repositioned after the last sibling NP.

CPs Relative clauses correspond to the CP category in the Treebank. Figure 7 shows an example noun phrase with two nested CP modifiers. As illustrated in the figure, relative clauses in Chinese also occur before the noun they modify, which makes the word order of this sentence quite different from that of the English translation. Such distortions in the word reordering will be quite difficult for the word or phrase-based alignment model to capture. However, with the application of a reordering rule to reposition the child CP after its sibling NP under a parent NP, and the PP VP reordering rule for VP introduced previously, the sentence can be easily transformed into "*French delegation participate 8th handicap people Winter Olympics hold at US Salt Lake City*," a sentence whose word order is much closer to that of English.

CP is typically formed by "IP+DEC", in which DEC's only function is to mark the IP as a relative clause, similar to the function of "that" in English. We use a rule to bring DEC to the front of IP under CP, to make it more aligned with the "that + clause" structure of English.

```
NP CP IP VP VV 参加 (participate)
            NP CP IP VP PP P 在 (at)
                           NP NR 美国(US)
                              NR 盐湖城
                        (Salt Lake City)
                        VP VV 举行 (hold)
                  DEC 的 (DEC)
               QP OD 第八 (8th)
                  CLP M 届 (measure word)
               NPB NN 残疾人
                      (handicap people)
                   NR 冬奥会
                      (Winter Olympics)
      DEC 的 (DEC)
   NPB NR 法国 (French)
   NPB NN 代表队 (delegation)
```

Figure 7: An example with two nested CP modifiers. The phrase translates into "*the French delegation participating in the 8th Special Winter Olympics held in Salt Lake City US*."

```
LCP IP NP-A NPB NN 事故(accident)
       VP VV 发生(happen)
    LC 后(after)
```

Figure 8: An example Chinese localizer phrase. The phrase translates into "*after the accident happened*" in English.

3.3 Localizers

Figure 8 shows an example phrase of the type LCP. Localizers (tagged LC in the Treebank) in Chinese can be thought of as a post-phrasal preposition which is often used with temporal and locative phrases or clauses to mark directional information. They function similarly to prepositions and conjunctions in English such as "before," "on," "when," etc. Constituents of type LCP have a similar function to prepositional phrases. Sometimes they are combined with a pre-phrasal generic preposition "在" (roughly corresponding to "at" in English) to form a PP explicitly. An example is shown in Figure 9.

We developed a simple reordering rule which moves an LC node to immediately before its left sibling under a parent LCP node. This will result in a word order that is more similar to that of the English

```
PP P 在(at)
   LCP IP NP-A NPB NN 事故(accident)
          VP VV 发生(happen)
       LC 后(after)
```

Figure 9: An example Chinese PP encompassing an LCP. The phrase translates into "*after the accident happened*" in English.

prepositional phrase: the example in Figure 8 has the paraphrase "*after accident happen*" after the reordering rule is applied. In the case where an LCP is embedded in a parent PP phrase, the LC reordering rule will essentially merge the post-phrasal localizer with the pre-phrasal preposition. For example, the phrase in Figure 9 becomes "*at after accident happen*" after reordering. The phrase-based SMT system will have little problem in learning that "at after" translates into "after" in English.

4 Evaluation

Our baseline is a phrase-based MT system trained using the MOSES toolkit (Koehn et al., 2007). The training data consists of nearly 637K pairs of sentences from various parallel news corpora distributed by the Linguistic Data Consortium (LDC).[2] For tuning and testing, we use the official NIST MT evaluation data for Chinese from 2002 to 2006, which have four human generated English reference translations for each Chinese input. The evaluation data from 2002 to 2005 were split into two sets of roughly equal sizes: a tuning set of 2347 sentences is used for optimizing various parameters using minimum error training (also using the MOSES toolkit), and a development set of 2320 sentences is used for various analysis experiments. We report results on the NIST 2006 evaluation data.

A series of processing steps are needed before the reordering rules can be applied, which include segmentation, part-of-speech tagging, and parsing. We trained a Chinese Treebank-style tokenizer and part-of-speech tagger, both using a tagging model based on a perceptron learning algorithm (Collins, 2002). We used the Chinese parser described by Sun and Jurafsky (2004), which was adapted from the parser presented in Collins (1997). We then applied the reordering rules described in the previous section to the parse tree of each input. The reordered sentence is then re-tokenized to be consistent with the baseline system, which uses a different tokenization scheme that is more friendly to the MT system.[3]

	Dev	Nist06
Baseline	31.57	28.52
Reorder	32.86	30.86
Gain	+1.29	+2.34

Table 2: BLEU score of the baseline and reordered systems.

We use BLEU scores as the performance measure in our evaluation (Papineni et al., 2002). Table 2 gives results for the baseline and reordered systems on both the development and test sets. As shown in the table, the reordering method is able to improve the BLEU scores by 1.29 points on the development set, and by 2.34 on the NIST 2006 set.

4.1 Frequency and Accuracy of Reordering Rules

We collected statistics to evaluate how often and accurately the reordering rules are applied in the data. The accuracy is measured in terms of the percentage of rule applications that correctly reorder sentences. The vast majority of reordering errors are due to parsing mistakes.

Table 3 summarizes the count of each rule in the training data, ignoring rules occurring less than 500 times in the training data, and the number of sentences each rule impacts. The most frequent three rules are NP(CP:NP), VP(PP:VP), and DNP(NP):NP, which account for over 76% of all the reordering instances and jointly affect 74% of all the training sentences. This shows the prevalence of systematic word order differences between Chinese and English. Only 122,076 (or 19.2%) sentences remain unchanged after the reordering rules are applied.

Each of the processing steps in producing the Chinese parse tree is prone to error and could lead to mistakes in the reordering of the Chinese sentence.

[2] We used 8 corpora for training, including LDC2002E18, LDC2003E07, LDC2003E14, LDC2005E83, LDC2005T06, LDC2006E26, LDC2006E8, and LDC2006G05.

[3] The tokenizer used by the MT system favors smaller word units, and backs off to a character by character scheme for unknown words.

Type	Rule Name	Counts	# Sent.
VP	VP(PP:VP)	331,827	258,214
	VP(NT:VP)	23,353	22,926
	VP(LCP:VP)	8,674	8,661
	VP(QP:VP)	7,834	7,777
NP	NP(CP:NP)	345,165	262,588
	DNP(NP):NP	280,367	218,865
	DNP(PP):NP	38,225	36,295
	DNP(LCP):NP	15,801	15,253
LC	LCP(NP:LC)	146,784	12,8333
	LCP(IP:LC)	36,923	35,749
	LCP(QP:LC)	14,893	14,287
Total		1,249,846	636,686

Table 3: Statistics of various reordering rules in the training data.

To assess the accuracy of reordering rules, we conducted human evaluations on a set of 200 sentences randomly selected from the development set. Within this set, there were in total 155 sentences containing at least one reordering rule, with 339 rules in total. A bilingual speaker was presented with the Chinese parse tree, the sentence before and after the reordering, and the particular reordering rules applied to the sentence. The bilingual rater determined the correctness of each rule by first identifying the scope of the rule and comparing the string before and after reordering, referencing the corresponding parse structure if necessary. Table 4 summarizes the accuracy (precision) for each type of rule. Notice that our human evaluation of the reordering rules does not take into account missed reordering.

Overall, there are a lot of reordering errors caused by incorrect parses. On a sentence level, only 57 out of the 155 reordered sentences (36.8%) are error free. Nevertheless, syntactic reordering seems to be helpful in improving the translation quality, despite noise introduced into the data due to the errors.

4.2 Impact of Individual Reordering Rules

In order to assess the relative effectiveness of the reordering rules, we conducted an experiment in which we trained and tested systems using data that were reordered using different subsets of the reordering rules. Table 5 summarizes the BLEU scores of the reordered system for each rule type.

	Count	Accuracy
VP rules	108	65.7%
NP rules	209	54.6%
LC rules	76	77.6%
All rules	393	62.1%

Table 4: Accuracy of reordering rules on a set of 200 sentences randomly selected from the development set.

	BLEU	Gain
Baseline	31.57	-
VP rules	32.71	+1.14
NP rules	32.23	+0.66
LC rules	31.59	+0.02
All rules	32.86	+1.29

Table 5: Comparison of translation performance with different types of reordering rules. *Gain* is the change in BLEU score when compared to the baseline system. All results are on the development set.

As shown in the table, the VP rules are more effective than the NP rules, even though the NP rules are more frequent than the VP rules in the data. This is perhaps because the reordering of VP modifiers achieves a slightly higher accuracy than that of the NP modifiers. We are a bit surprised by the lack of performance gains with the LC rules only. More analysis is needed to explain this behavior.

4.3 Better Alignment?

There could be two reasons why the syntactic reordering approach improves over the baseline phrase-based SMT system. One obvious benefit is that the word order of the transformed source sentence is much closer to that of the target sentence, which reduces the reliance on the distortion model to perform reordering during decoding. Another potential benefit is that the alignment between the two sides will be of higher quality because of fewer “distortions” between the source and the target, so that the resulting phrase table of the reordered system would be better. However, a counter argument is that the reordering is very error prone, so that the added noise in the reordered data would actually hurt the alignments and hence the phrase table.

Lacking a good way to measure the quality of

	Original Dev	Reordered Dev
Baseline	31.57	32.19
Reorder	30.67	32.86

Table 6: Comparison of BLEU scores in matched and mismatched conditions. The baseline and reordered systems were first tuned on mismatched data before being tested on mismatched data.

the phrase table directly, we conducted an experiment in which we tested the baseline and reordered systems with both the original and reordered development data. The idea is to compare the two systems given the same type of input: if the reordered system learned a better phrase table, then it might outperform the baseline system on un-reordered inputs despite the mismatch; on the other hand, if the baseline system learned a better phrase table, then it might outperform the reordered system on reordered inputs despite the mismatch. However, the results in Table 6 did not settle our question: the reordered system performed worse than the baseline on unreordered data, while the baseline system performed worse than the reordered system on reordered data, both of which can be explained by the mismatched conditions between training and testing. Perhaps more interesting is the performance gap of the baseline system on the reordered data vs. on the original data: it achieved 0.62 BLEU score gain despite the mismatch in training and testing conditions.

5 Discussion and Future Work

In this paper, we described a set of syntactic reordering rules that exploit systematic differences between Chinese and English word order to transform Chinese sentences to be much closer to English in terms of their word order. We evaluated the reordering approach within the MOSES phrase-based SMT system (Koehn et al., 2007). The reordering approach improved the BLEU score for the MOSES system from 28.52 to 30.86 on the NIST 2006 evaluation data. Our manual evaluation of the reordering accuracy indicated that the reordering approach is helpful at improving the translation quality despite relatively frequent reordering errors. The reordering approach even achieved a 0.62 gain in BLEU score when only the test data are reordered.

An important category we examined but did not reorder was clauses of type `IP`, which generally corresponds to declarative sentences in Chinese. Sentences of this form have quite similar top-level constituent ordering to English: both follow SVO (subject-verb-object) order. There are several special cases in which English and Chinese differ, the most notable being the topicalization of objects or temporal and locative noun phrases (which function as adverbial phrases). We did not try to restore them to the canonical order for several reasons. First, topicalization of temporal and locative phrases happens in English as well. For example, “In Israel yesterday, an explosion killed one person and injured twelve” is a perfectly acceptable English sentence. Second, the parser’s performance on special constructions is likely to be poor, resulting in frequent reordering errors. Third, special constructions that do not occur often in the data are less likely to have a significant impact on the translation performance. Thus our strategy has been to find reordering rules for syntactic categories that are common in the data and systematically different between the two languages.

In our experiments, the phrase-based MT system uses an un-lexicalized reordering model, which might make the effects of the syntactic reordering method more pronounced. However, in an early experiment[4] submitted to the official NIST 2006 MT evaluation, the reordered system also improved the BLEU score substantially (by 1.34 on NIST 2006 data) over a phrase-based MT system with lexicalized reordering models (Koehn et al., 2005). The same set of reordering rules in the experimental setting in the current paper achieve a 1.82 BLEU improvement on the same data set, which is comparable to the 1.34 gain for the lexicalized system.

We plan to output reordered lattices in the future, so that the approach would be more robust to errors made during parsing/reordering.

Acknowledgements

We would like to thank Brooke Cowan, Stephanie Seneff, and the three anonymous reviewers for their valuable comments. Thanks to Yushi Xu for evaluating the accuracy of the reordering rules. This work

[4] This experiment made use of a subset of the reordering rules we have presented here.

was supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–69.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John D. Lafferty, and Robert L. Mercer. 1992. Analysis, statistical transfer, and synthesis in machine translation. In *Proceedings of Conference on Theoretical and Methodological Issues in Machine Translation*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Poceedings of ACL*, pages 531–540.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of ACL*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Yuan Ding and Martha Palmer. 2005. Machine translation using probablistic synchronous dependency insertion grammars. In *Proceedings of ACL*, pages 541–548, Ann Arbor, Michigan.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, and Chris Callison-Burch. 2005. Edinburgh system description. In *IWSLT Speech Translation Evaluation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constrantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demonstration Session*.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of HLT-EMNLP*.

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of Coling 2004*, pages 625–630, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL*, pages 609–616.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of EMNLP*, pages 44–52, Sydney, Australia.

Sonja Niessen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of ACL*, pages 271–279, Ann Arbor, Michigan.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of Chinese. In *Proceedings of NAACL-HLT*.

Christoph Tillmann. 2004. A block orientation model for statistical machine translation. In *Proceedings of HLT-NAACL*, Boston, MA, USA.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of COLING*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL*.

Binarizing Syntax Trees to Improve Syntax-Based Machine Translation Accuracy

Wei Wang and **Kevin Knight** and **Daniel Marcu**
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA, 90292
{wwang,kknight,dmarcu}@languageweaver.com

Abstract

We show that phrase structures in Penn Treebank style parses are not optimal for syntax-based machine translation. We exploit a series of binarization methods to restructure the Penn Treebank style trees such that syntactified phrases smaller than Penn Treebank constituents can be acquired and exploited in translation. We find that by employing the EM algorithm for determining the binarization of a parse tree among a set of alternative binarizations gives us the best translation result.

1 Introduction

Syntax-based translation models (Eisner, 2003; Galley et al., 2006; Marcu et al., 2006) are usually built directly from Penn Treebank (PTB) (Marcus et al., 1993) style parse trees by composing treebank grammar rules. As a result, often no substructures corresponding to partial PTB constituents are extracted to form translation rules.

Syntax translation models acquired by composing treebank grammar rules assume that long rewrites are not decomposable into smaller steps. This effectively restricts the generalization power of the induced model. For example, suppose we have an xRs (Knight and Graehl, 2004) rule R_1 in Figure 1 that translates the Chinese phrase RUSSIA MINISTER VIKTOR-CHERNOMYRDIN into an English NPB tree fragment yielding an English phrase. Also suppose that we want to translate a Chinese phrase

VIKTOR-CHERNOMYRDIN AND HIS COLLEAGUE

into English. What we desire is that if we have another rule R_2 as shown in Figure 1, we could somehow compose it with R_1 to obtain the desirable translation. We unfortunately cannot do this because R_1 and R_2 are not further decomposable and their substructures cannot be re-used. The requirement that all translation rules have exactly one root node does not enable us to use the translation of VIKTOR-CHERNOMYRDIN in any other contexts than those seen in the training corpus.

A solution to overcome this problem is to right-binarize the left-hand side (LHS) (or the English-side) tree of R_1 such that we can decompose R_1 into R_3 and R_4 by factoring `NNP(viktor) NNP(chernomyrdin)` out as R_4 according to the word alignments; and left-binarize the LHS of R_2 by introducing a new tree node that collapses the two NNP's, so as to generalize this rule, getting rule R_5 and rule R_6. We also need to consistently syntactify the root labels of R_4 and the new frontier label of R_6 such that these two rules can be composed. Since labeling is not a concern of this paper, we simply label new nodes with X-bar where X here is the parent label. With all these in place, we now can translate the foreign sentence by composing R_6 and R_4 in Figure 1.

Binarizing the syntax trees for syntax-based machine translation is similar in spirit to generalizing parsing models via markovization (Collins, 1997; Charniak, 2000). But in translation modeling, it is unclear how to effectively markovize the translation rules, especially when the rules are complex like those proposed by Galley et al. (2006).

In this paper, we explore the generalization ability of simple binarization methods like left-, right-, and head-binarization, and also their combinations. Simple binarization methods binarize syntax trees in a consistent fashion (left-, right-, or head-) and

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 746–754, Prague, June 2007. ©2007 Association for Computational Linguistics

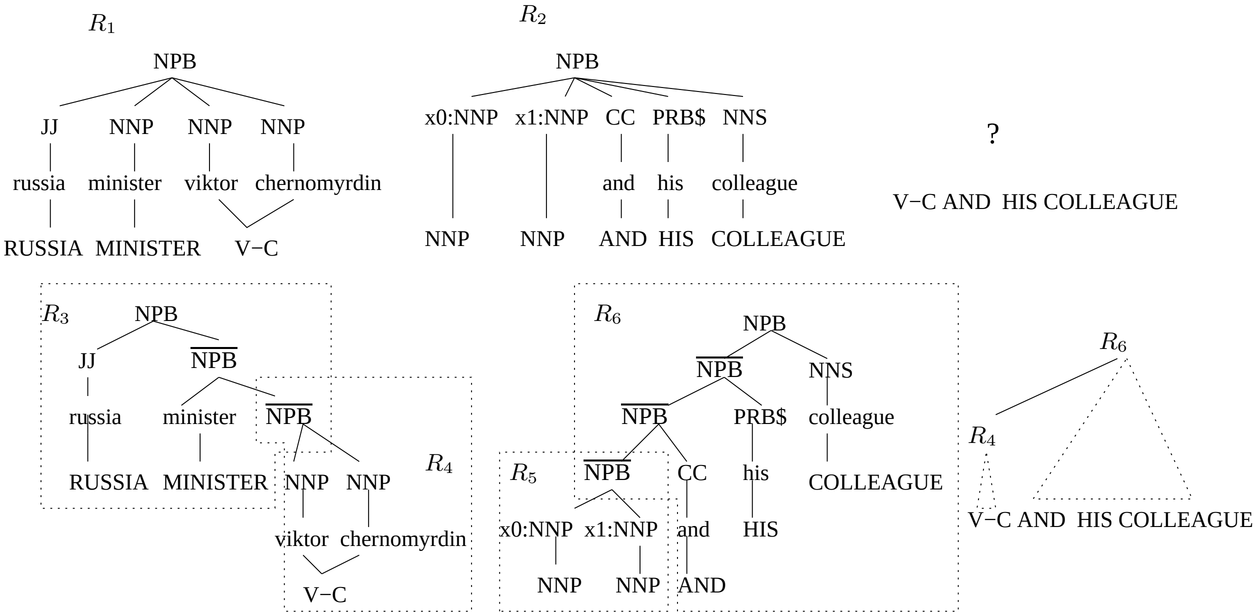


Figure 1: Generalizing translation rules by binarizing trees.

thus cannot guarantee that all the substructures can be factored out. For example, right binarization on the LHS of R_1 makes available R_4, but misses R_6 on R_2. We then introduce a *parallel restructuring* method, that is, one can binarize both to the left and right at the same time, resulting in a binarization forest. We employ the EM (Dempster et al., 1977) algorithm to learn the binarization bias for each tree node from the parallel alternatives. The EM-binarization yields best translation performance.

The rest of the paper is organized as follows. Section 2 describes related research. Section 3 defines the concepts necessary for describing the binarizations methods. Section 4 describes the tree binarization methods in details. Section 5 describes the forest-based rule extraction algorithm, and section 6 explains how we restructure the trees using the EM algorithm. The last two sections are for experiments and conclusions.

2 Related Research

Several researchers (Melamed et al., 2004; Zhang et al., 2006) have already proposed methods for binarizing synchronous grammars in the context of machine translation. Grammar binarization usually maintains an equivalence to the original grammar such that binarized grammars generate the same language and assign the same probability to each string as the original grammar does. Grammar binarization is often employed to make the grammar fit in a CKY parser. In our work, we are focused on binarization of parse trees. Tree binarization generalizes the resulting grammar and changes its probability distribution. In tree binarization, synchronous grammars built from restructured (binarized) training trees still contain non-binary, multi-level rules and thus still require the binarization transformation so as to be employed by a CKY parser.

The translation model we are using in this paper belongs to the xRs formalism (Knight and Graehl, 2004), which has been proved successful for machine translation in (Galley et al., 2004; Galley et al., 2006; Marcu et al., 2006).

3 Concepts

We focus on tree-to-string (in noisy-channel model sense) translation models. Translation models of this type are typically trained on tuples of a source-language sentence **f**, a target language (e.g., English) parse tree π that yields $\mathbf{e}$ and translates from **f**, and the word alignments **a** between **e** and **f**. Such a tuple is called an alignment graph in (Galley et al., 2004). The graph (1) in Figure 2 is such an alignment graph.

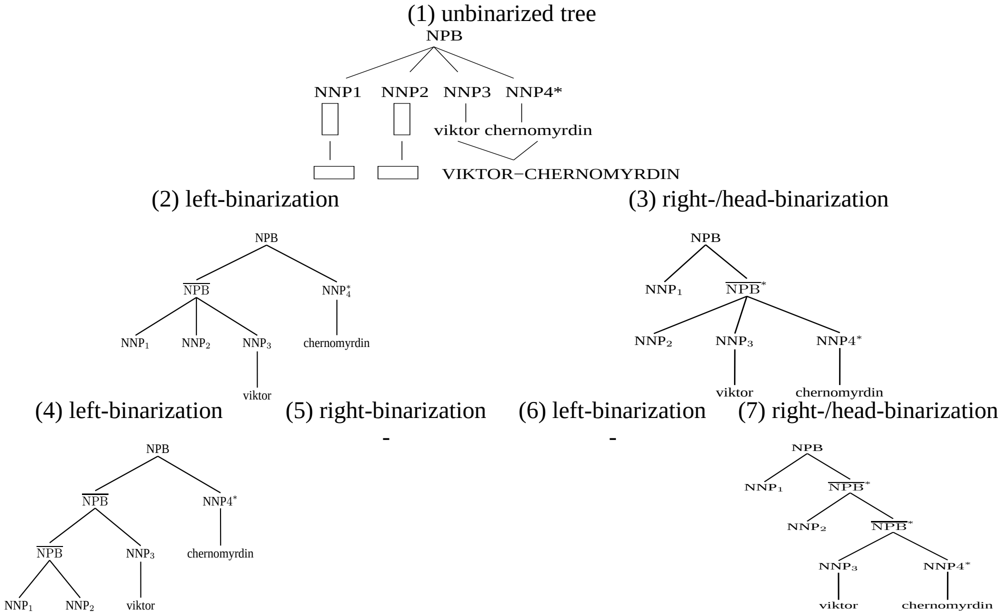


Figure 2: Left, right, and head binarizations. Heads are marked with *'s. New nonterminals introduced by binarization are denoted by X-bars.

A tree node in π is *admissible* if the **f** string covered by the node is contiguous but not empty, and if the **f** string does not align to any **e** string that is not covered by π. An xRs rule can be extracted only from an admissible tree node, so that we do not have to deal with dis-contiguous **f** spans in decoding (or synchronous parsing). For example, in tree (2) in Figure 2, node $\overline{\text{NPB}}$ is not admissible because the **f** string that the node covers also aligns to NNP4, which is not covered by the $\overline{\text{NPB}}$. Node $\overline{\text{NPB}}$ in tree (3), on the other hand, is admissible.

A set of sibling tree nodes is called *factorizable* if we can form an admissible new node dominating them. For example, in tree (1) of Figure 2, sibling nodes NNP$_2$ NNP$_3$ and NNP$_4$ are factorizable because we can factorize them out and form a new node $\overline{\text{NPB}}$, resulting in tree (3). Sibling tree nodes NNP$_1$ NNP$_2$ and NNP$_3$ are not factorizable. In synchronous parse trees, not all sibling nodes are factorizable, thus not all sub-phrases can be acquired and syntactified. The main purpose of our paper is to restructure parse trees by factorization such that syntactified sub-phrases can be employed in translation.

4 Binarizing Syntax Trees

We are going to binarize a tree node n that dominates r children n_1, ..., n_r. Restructuring will be performed by introducing new tree nodes to dominate a subset of the children nodes. To avoid overgeneralization, we allow ourselves to form only one new node at a time. For example, in Figure 2, we can binarize tree (1) into tree (2), but we are not allowed to form two new nodes, one dominating NNP$_1$ NNP$_2$ and the other dominating NNP$_3$ NNP$_4$. Since labeling is not the concern of this paper, we relabel the newly formed nodes as $\overline{n}$.

4.1 Simple binarization methods

The *left binarization* of node n (i.e., the NPB in tree (1) of Figure 2) factorizes the leftmost $r-1$ children by forming a new node $\overline{n}$ (i.e., $\overline{\text{NPB}}$ in tree (2)) to dominate them, leaving the last child n_r untouched; and then makes the new node $\overline{n}$ the left child of n. The method then recursively left-binarizes the newly formed node $\overline{n}$ until two leaves are reached. In Figure 2, we left-binarize tree (1) into (2) and then into (4).

The *right binarization* of node n factorizes the rightmost $r-1$ children by forming a new node $\overline{n}$ (i.e., $\overline{\text{NPB}}$ in tree (3)) to dominate them, leaving the first child n_1 untouched; and then makes the new node $\overline{n}$ the right child of n. The method then recursively right-binarizes the newly formed node $\overline{n}$. In Figure 2, we right-binarize tree (1) into (3) and then into (7).

The *head binarization* of node n left-binarizes n if the head is the first child; otherwise, right-binarizes n. We prefer right-binarization to left-binarization when both are applicable under the head restriction because our initial motivation was to generalize the NPB-rooted translation rules. As we will show in the experiments, binarization of other types of phrases contribute to the translation accuracy improvement as well.

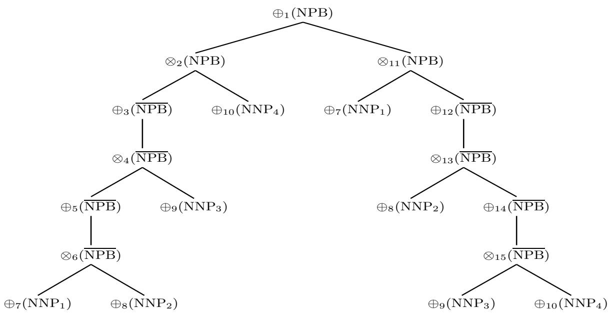
Any of these simple binarization methods is easy to implement, but is incapable of giving us all the factorizable sub-phrases. Binarizing all the way to the left, for example, from tree (1) to tree (2) and to tree (4) in Figure 2, does not enable us to acquire a substructure that yields NNP$_3$ NNP$_4$ and their translational equivalences. To obtain more factorizable sub-phrases, we need to *parallel-binarize* in both directions.

4.2 Parallel binarization

Simple binarizations transform a parse tree into another single parse tree. Parallel binarization will transform a parse tree into a binarization forest, desirably packed to enable dynamic programming when extracting translation rules from it.

Borrowing terms from parsing semirings (Goodman, 1999), a packed forest is composed of additive forest nodes ($\oplus$-nodes) and multiplicative forest nodes ($\otimes$-nodes). In the binarization forest, a $\otimes$-node corresponds to a tree node in the unbinarized tree; and this $\otimes$-node composes several $\oplus$-nodes, forming a one-level substructure that is observed in the unbinarized tree. A $\oplus$-node corresponds to alternative ways of binarizing the same tree node in the unbinarized tree and it contains one or more $\otimes$-nodes. The same $\oplus$-node can appear in more than one place in the packed forest, enabling sharing. Figure 3 shows a packed forest obtained by packing trees (4) and (7) in Figure 2 via the following parallel binarization algorithm.

To parallel-binarize a tree node n that has children $n_1, ..., n_r$, we employ the following steps:



$\oplus_1$(NPB)
$\otimes_2$(NPB)
$\otimes_{11}$(NPB)
$\oplus_3(\overline{\text{NPB}})$
$\oplus_{10}$(NNP$_4$)
$\oplus_7$(NNP$_1$)
$\oplus_{12}(\overline{\text{NPB}})$
$\otimes_4(\overline{\text{NPB}})$
$\otimes_{13}(\overline{\text{NPB}})$
$\oplus_5(\overline{\text{NPB}})$
$\oplus_9$(NNP$_3$)
$\oplus_8$(NNP$_2$)
$\oplus_{14}(\overline{\text{NPB}})$
$\otimes_6(\overline{\text{NPB}})$
$\otimes_{15}(\overline{\text{NPB}})$
$\oplus_7$(NNP$_1$)
$\oplus_8$(NNP$_2$)
$\oplus_9$(NNP$_3$)
$\oplus_{10}$(NNP$_4$)

Figure 3: Packed forest obtained by packing trees (4) and (7) in Figure 2

- We recursively parallel-binarize children nodes n_1, ..., n_r, producing binarization $\oplus$-nodes $\oplus(n_1)$, ..., $\oplus(n_r)$, respectively.

- We right-binarize n, if any contiguous[1] subset of children $n_2, ..., n_r$ is factorizable, by introducing an intermediate tree node labeled as $\overline{n}$. We recursively parallel-binarize $\overline{n}$ to generate a binarization forest node $\oplus(\overline{n})$. We form a multiplicative forest node $\otimes_R$ as the parent of $\oplus(n_1)$ and $\oplus(\overline{n})$.

- We left-binarize n if any contiguous subset of $n_1, ..., n_{r-1}$ is factorizable and if this subset contains n_1. Similar to the above right-binarization, we introduce an intermediate tree node labeled as $\overline{n}$, recursively parallel-binarize $\overline{n}$ to generate a binarization forest node $\oplus(\overline{n})$, form a multiplicative forest node $\otimes_L$ as the parent of $\oplus(\overline{n})$ and $\oplus(n_1)$.

- We form an additive node $\oplus(n)$ as the parent of the two already formed multiplicative nodes $\otimes_L$ and $\otimes_R$.

The (left and right) binarization conditions consider any subset to enable the factorization of small constituents. For example, in tree (1) of Figure 2, although NNP$_1$ NNP$_2$ NNP$_3$ of NPB are not factorizable, the subset NNP$_1$ NNP$_2$ is factorizable. The binarization from tree (1) to tree (2) serves as a relaying step for us to factorize NNP$_1$ NNP$_2$ in tree (4). The left-binarization condition is stricter than

[1] We factorize only subsets that cover contiguous spans to avoid introducing dis-contiguous constituents for practical purpose. In principle, the algorithm works fine without this binarization condition.

the right-binarization condition to avoid spurious binarization; i.e., to avoid the same subconstituent being reached via both binarizations. We could transform tree (1) directly into tree (4) without bothering to generate tree (3). However, skipping tree (3) will create us difficulty in applying the EM algorithm to choose a better binarization for each tree node, since tree (4) can neither be classified as left binarization nor as right binarization of the original tree (1) — it is the result of the composition of two left-binarizations.

In parallel binarization, nodes are not always binarizable in both directions. For example, we do not need to right-binarize tree (2) because NNP$_2$ NNP$_3$ are not factorizable, and thus cannot be used to form sub-phrases. It is still possible to right-binarize tree (2) without affecting the correctness of the parallel binarization algorithm, but that will spuriously increase the branching factor of the search for the rule extraction, because we will have to expand more tree nodes.

A restricted version of parallel binarization is the *headed parallel binarization*, where both the left and the right binarization must respect the head propagation property at the same time.

A nice property of parallel binarization is that for any factorizable substructure in the unbinarized tree, we can always find a corresponding admissible $\oplus$-node in the parallel-binarized packed forest. A leftmost substructure like the lowest $\overline{\text{NPB}}$-subtree in tree (4) of Figure 2 can be made factorizable by several successive left binarizations, resulting in $\oplus_5(\overline{\text{NPB}})$-node in the packed forest in Figure 3. A substructure in the middle can be factorized by the composition of several left- and right-binarizations. Therefore, after a tree is parallel-binarized, to make the sub-phrases available to the MT system, all we need to do is to extract rules from the admissible nodes in the packed forest. Rules that can be extracted from the original unrestructured tree can be extracted from the packed forest as well.

Parallel binarization results in parse forests. Thus translation rules need to be extracted from training data consisting of (e-forest, **f**, **a**)-tuples.

5 Extracting translation rules from (e-forest, f, a)-tuples

The algorithm to extract rules from (e-forest, **f**, **a**)-tuples is a natural generalization of the (e-parse, **f**, **a**)-based rule extraction algorithm in (Galley et al., 2006). The input to the forest-based algorithm is a (e-forest, **f**, **a**)-triple. The output of the algorithm is a derivation forest (Galley et al., 2006) composed of xRs rules. The algorithm recursively traverses the e-forest top-down and extracts rules only at admissible forest nodes.

The following procedure transforms the packed e-forest in Figure 3 into a packed synchronous derivation in Figure 4.

Condition 1: Suppose we reach an additive e-forest node, e.g. $\oplus_1(\text{NPB})$ in Figure 3. For each of $\oplus_1(\text{NPB})$'s children, e-forest nodes $\otimes_2(\text{NPB})$ and $\otimes_{11}(\text{NPB})$, we go to condition 2 to recursively extract rules on these two e-forest nodes, generating multiplicative derivation forest nodes, i.e., $\otimes$(NPB($\overline{\text{NPB}}$: x_0 NNP$_3$(viktor) NNP$_4$(chernomyrdin)$_4$) $\rightarrow$ x_0 V-C) and $\otimes$(NPB(NNP$_1$ $\overline{\text{NPB}}$(NNP$_2$: x_0 $\overline{\text{NPB}}$: x_1)) $\rightarrow$ x_0 x_1 x_2) in Figure 4. We make these new $\otimes$ nodes children of $\oplus(\text{NPB})$ in the derivation forest.

Condition 2: Suppose we reach a multiplicative parse forest node, i.e., $\otimes_{11}(\text{NPB})$ in Figure 3. We extract rules rooted at it using the procedure in (Galley et al., 2006), forming multiplicative derivation forest nodes, i.e., $\otimes$(NPB(NNP$_1$ $\overline{\text{NPB}}$(NNP$_2$: x_0 $\overline{\text{NPB}}$: x_1)) $\rightarrow$ x_0 x_1 x_2) We then go to condition 1 to form the derivation forest on the additive frontier e-forest nodes of the newly extracted rules, generating additive derivation forest nodes, i.e., $\oplus$(NNP$_1$), $\oplus$(NNP$_2$) and $\oplus(\overline{\text{NPB}})$. We make these $\oplus$ nodes the children of node $\otimes$(NPB(NNP$_1$ $\overline{\text{NPB}}$(NNP$_2$: x_0 $\overline{\text{NPB}}$: x_1)) $\rightarrow$ x_0 x_1 x_2) in the derivation forest.

This algorithm is a natural extension of the extraction algorithm in (Galley et al., 2006) in the sense that we have an extra condition (1) to relay rule extraction on additive e-forest nodes.

It is worthwhile to eliminate the spuriously ambiguous rules that are introduced by the parallel bi-

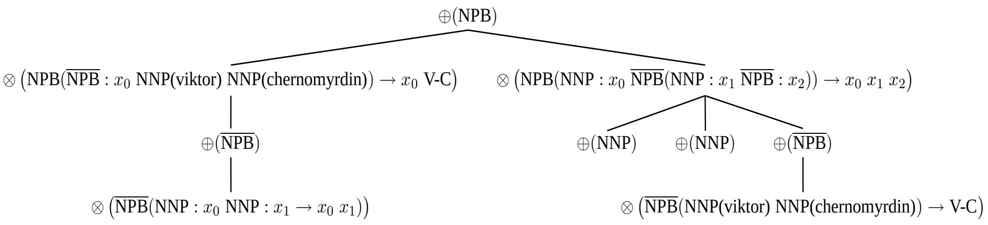

⊕(NPB)
⊗ (NPB($\overline{\mathrm{NPB}}$: x_0 NNP(viktor) NNP(chernomyrdin)) → x_0 V-C)
⊗ (NPB(NNP : x_0 $\overline{\mathrm{NPB}}$(NNP : x_1 $\overline{\mathrm{NPB}}$: x_2)) → x_0 x_1 x_2)
⊕($\overline{\mathrm{NPB}}$)
⊕(NNP)
⊕(NNP)
⊕($\overline{\mathrm{NPB}}$)
⊗ ($\overline{\mathrm{NPB}}$(NNP : x_0 NNP : x_1 → x_0 x_1))
⊗ ($\overline{\mathrm{NPB}}$(NNP(viktor) NNP(chernomyrdin)) → V-C)

Figure 4: Derivation forest.

narization. For example, we may extract the following two rules:

- $\overline{\mathrm{A}}(\overline{\mathrm{A}}(\mathrm{B}{:}x_0\ \mathrm{C}{:}x_1)\mathrm{D}{:}x_2) \longrightarrow x_1\ x_0\ x_2$
- $\overline{\mathrm{A}}(\mathrm{B}{:}x_0\ \overline{\mathrm{A}}(\mathrm{C}{:}x_1\ \mathrm{D}{:}x_2)) \longrightarrow x_1\ x_0\ x_2$

These two rules, however, are not really distinct. They both converge to the following rules if we delete the auxiliary nodes $\overline{A}$.

- $\overline{\mathrm{A}}(\mathrm{B}{:}x_0\ C{:}x_1\ D{:}x_2) \longrightarrow x_1\ x_0\ x_2$

The forest-base rule extraction algorithm produces much larger grammars than the tree-based one, making it difficult to scale to very large training data. From a 50M-word Chinese-to-English parallel corpus, we can extract more than 300 million translation rules, while the tree-based rule extraction algorithm gives approximately 100 million. However, the restructured trees from the simple binarization methods are not guaranteed to give the best trees for syntax-based machine translation. What we desire is a binarization method that still produces single parse trees, but is able to mix left binarization and right binarization in the same tree. In the following, we shall use the EM algorithm to learn the desirable binarization on the forest of binarization alternatives proposed by the parallel binarization algorithm.

6 Learning how to binarize via the EM algorithm

The basic idea of applying the EM algorithm to choose a restructuring is as follows. We perform a set $\{\beta\}$ of binarization operations on a parse tree τ. Each binarization β is the sequence of binarizations on the necessary (i.e., factorizable) nodes in τ in preorder. Each binarization β results in a restructured tree τ_β. We extract rules from (τ_β, **f**, **a**), generating a translation model consisting of parameters (i.e., rule probabilities) θ. Our aim is to obtain the binarization β^* that gives the best likelihood of the restructured training data consisting of (τ_β, **f** , **a**)-tuples. That is

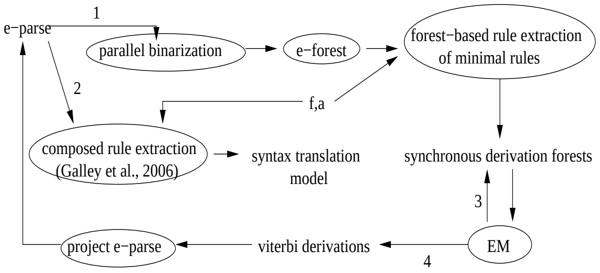


Figure 5: Using the EM algorithm to choose restructuring.

$$\beta^* \quad = \quad \arg\max_{\beta} p(\tau_\beta, \mathbf{f}, \mathbf{a}|\theta^*) \tag{1}$$

In practice, we cannot enumerate all the exponential number of binarized trees for a given e-parse. We therefore use the packed forest to store all the binarizations that operate on an e-parse in a compact way, and then use the inside-outside algorithm (Lari and Young, 1990; Knight and Graehl, 2004) for model estimation.

The probability $p(\tau_\beta, \mathbf{f}, \mathbf{a})$ of a (τ_β, **f**, **a**)-tuple is what the basic syntax-based translation model is concerned with. It can be further computed by aggregating the rule probabilities $p(r)$ in each derivation ω in the set of all derivations Ω (Galley et al., 2004; Marcu et al., 2006). That is

$$p(\tau_\beta, \mathbf{f}, \mathbf{a}) \quad = \quad \sum_{\omega \in \Omega} \prod_{r \in \omega} p(r) \tag{2}$$

Since it has been well-known that applying EM with tree fragments of different sizes causes overfitting (Johnson, 1998), and since it is also known that syntax MT models with larger composed rules in the mix significantly outperform rules that minimally explain the training data (minimal rules) in translation accuracy (Galley et al., 2006), we decompose $p(\tau_b, \mathbf{f}, \mathbf{a})$ using minimal rules during running of the EM algorithm, but, after the EM restructuring is finished, we build the final translation model using composed rules for evaluation.

Figure 5 is the actual pipeline that we use for EM binarization. We first generate a packed e-forest via parallel binarization. We then extract minimal translation rules from the (e-forest, **f**, **a**)-tuples, producing synchronous derivation forests. We run the inside-outside algorithm on the derivation forests until convergence. We obtain the Viterbi derivations and project the English parses from the derivations. Finally, we extract composed rules using Galley et al. (2006)'s (e-tree, **f**, **a**)-based rule extraction algorithm. This procedure corresponds to the path 13^*42 in the pipeline.

7 Experiments

We carried out a series of experiments to compare the performance of different binarization methods in terms of BLEU on Chinese-to-English translation tasks.

7.1 Experimental setup

Our bitext consists of 16M words, all in the mainland-news domain. Our development set is a 925-line subset of the 993-line NIST02 evaluation set. We removed long sentences from the NIST02 evaluation set to speed up discriminative training. The test set is the full 919-line NIST03 evaluation set.

We used a bottom-up, CKY-style decoder that works with binary xRs rules obtained via a synchronous binarization procedure (Zhang et al., 2006). The decoder prunes hypotheses using strategies described in (Chiang, 2007).

The parse trees on the English side of the bitexts were generated using a parser (Soricut, 2004) implementing the Collins parsing models (Collins, 1997).

We used the EM procedure described in (Knight and Graehl, 2004) to perform the inside-outside algorithm on synchronous derivation forests and to generate the Viterbi derivation forest.

We used the rule extractor described in (Galley et al., 2006) to extract rules from (e-parse, **f**, **a**)-tuples, but we made an important modification: new nodes introduced by binarization will not be counted when computing the rule size limit unless they appear as the rule roots. The motivation is that binarization deepens the parses and increases the number of tree nodes. In (Galley et al., 2006), a composed rule is extracted only if the number of internal nodes it contains does not exceed a limit (i.e., 4), similar to the phrase length limit in phrase-based systems. This means that rules extracted from the restructured trees will be smaller than those from the unrestructured trees, if the $\overline{X}$ nodes are deleted. As shown in (Galley et al., 2006), smaller rules lose context, and thus give lower translation performance. Ignoring $\overline{X}$ nodes when computing the rule sizes preserves the unstructured rules in the resulting translation model and adds substructures as bonuses.

7.2 Experiment results

Table 1 shows the BLEU scores of mixed-cased and detokenized translations of different systems. We see that all the binarization methods improve the baseline system that does not apply any binarization algorithm. The EM-binarization performs the best among all the restructuring methods, leading to 1.0 BLEU point improvement. We also computed the bootstrap p-values (Riezler and Maxwell, 2005) for the pairwise BLEU comparison between the baseline system and any of the system trained from binarized trees. The significance test shows that the EM binarization result is statistically significant better than the baseline system ($p > 0.005$), even though the baseline is already quite strong. To our best knowledge, 37.94 is the highest BLEU score on this test set to date.

Also as shown in Table 1, the grammars trained from the binarized training trees are almost two times of the grammar size with no binarization. The extra rules are substructures factored out by these binarization methods.

How many more substructures (or translation rules) can be acquired is partially determined by how many more admissible nodes each binarization method can factorize, since rules are extractable only from admissible tree nodes. According to Table 1, binarization methods significantly increase the number of admissible nodes in the training trees. The EM binarization makes available the largest

EXPERIMENT	NIST03-BLEU	# RULES	# ADMISSIBLE NODES IN TRAINING
no-bin	36.94	63.4M	7,995,569
left binarization	37.47 ($p = 0.047$)	114.0M	10,463,148
right binarization	37.49 ($p = 0.044$)	113.0M	10,413,194
head binarization	37.54 ($p = 0.086$)	113.8M	10,534,339
EM binarization	**37.94** ($p = 0.0047$)	115.6M	10,658,859

Table 1: Translation performance, grammar size and # admissible nodes versus binarization algorithms. BLEU scores are for mixed-cased and detokenized translations, as we usually do for NIST MT evaluations.

nonterminal	left-binarization	right-binarization
NP	**96.97**%	3.03%
NP-C	**97.49**%	2.51%
NPB	0.25%	**99.75**%
VP	**93.90**%	6.10%
PP	**83.75**%	16.25%
ADJP	**87.83**%	12.17%
ADVP	**82.74**%	17.26%
S	**85.91**%	14.09%
S-C	18.88%	**81.12**%
SBAR	**96.69**%	3.31%
QP	**86.40**%	13.60%
PRN	**85.18**%	14.82%
WHNP	**97.93**%	2.07%
NX	**100**%	0
SINV	**87.78**%	12.22%
PRT	**100**%	0
SQ	**93.53**%	6.47%
CONJP	18.08%	**81.92**%

Table 2: Binarization bias learned by EM.

number of admissible nodes, and thus results in the most rules.

The EM binarization factorizes more admissible nodes because it mixes both left and right binarizations in the same tree. We computed the binarization biases learned by the EM algorithm for each nonterminal from the binarization forest of headed-parallel binarizations of the training trees, getting the statistics in Table 2. Of course, the binarization bias chosen by left-/right-binarization methods would be 100% deterministic. One noticeable message from Table 2 is that most of the categories are actually biased toward left-binarization, although our motivating example in our introduction section is for NPB, which needed right binarization. The main reason might be that the head sub-constituents of most categories tend to be on the left, but according to the performance comparison between head binarization and EM binarization, head binarization does not suffice because we still need to choose the binarization between left and right if they both are head binarizations.

8 Conclusions

In this paper, we not only studied the impact of simple tree binarization algorithms on the performance of end-to-end syntax-based MT, but also proposed binarization methods that mix more than one simple binarization in the binarization of the same parse tree. Binarizing a tree node whether to the left or to the right was learned by employing the EM algorithm on a set of alternative binarizations and by choosing the Viterbi one. The EM binarization method is informed by word alignments such that unnecessary new tree nodes will not be “blindly” introduced.

To our best knowledge, our research is the first work that aims to generalize a syntax-based translation model by restructuring and achieves significant improvement on a strong baseline. Our work differs from traditional work on binarization of synchronous grammars in that we are not concerned with the equivalence of the binarized grammar to the original grammar, but intend to generalize the original grammar via restructuring of the training parse trees to improve translation performance.

Acknowledgments

The authors would like to thank David Chiang, Bryant Huang, and the anonymous reviewers for their valuable feedbacks.

References

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Seattle, May.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the*

35th Annual Meeting of the Association for Computational Linguistics (ACL), pages 16–23, Madrid, Spain, July.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 205–208, Sapporo, July.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a Translation Rule? In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, Boston, Massachusetts.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Models. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

M. Johnson. 1998. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76.

K. Knight and J. Graehl. 2004. Training Tree Transducers. In *Proceedings of NAACL-HLT*.

K. Lari and S. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, pages 35–56.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phraases. In *Proceedings of EMNLP-2006, pp. 44-52*, Sydney, Australia.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proc. ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

Radu Soricut. 2004. A reimplementation of Collins's parsing models. Technical report, Information Sciences Institute, Department of Computer Science University of Southern California.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the HLT-NAACL*.

What Can Syntax-based MT Learn from Phrase-based MT?

Steve DeNeefe and Kevin Knight
Information Sciences Institute
The Viterbi School of Engineering
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{sdeneefe,knight}@isi.edu

Wei Wang and Daniel Marcu
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA 90292
{wwang,dmarcu}@languageweaver.com

Abstract

We compare and contrast the strengths and weaknesses of a syntax-based machine translation model with a phrase-based machine translation model on several levels. We briefly describe each model, highlighting points where they differ. We include a quantitative comparison of the phrase pairs that each model has to work with, as well as the reasons why some phrase pairs are not learned by the syntax-based model. We then evaluate proposed improvements to the syntax-based extraction techniques in light of phrase pairs captured. We also compare the translation accuracy for all variations.

1 Introduction

String models are popular in statistical machine translation. Approaches include word substitution systems (Brown et al., 1993), phrase substitution systems (Koehn et al., 2003; Och and Ney, 2004), and synchronous context-free grammar systems (Wu and Wong, 1998; Chiang, 2005), all of which train on string pairs and seek to establish connections between source and target strings. By contrast, explicit syntax approaches seek to directly model the relations learned from parsed data, including models between source trees and target trees (Gildea, 2003; Eisner, 2003; Melamed, 2004; Cowan et al., 2006), source trees and target strings (Quirk et al., 2005; Huang et al., 2006), or source strings and target trees (Yamada and Knight, 2001; Galley et al., 2004).

It is unclear which of these important pursuits will best explain human translation data, as each has advantages and disadvantages. A strength of phrase models is that they can acquire all phrase pairs consistent with computed word alignments, snap those phrases together easily by concatenation, and re-order them under several cost models. An advantage of syntax-based models is that outputs tend to be syntactically well-formed, with re-ordering influenced by syntactic context and function words introduced to serve specific syntactic purposes.

A great number of MT models have been recently proposed, and other papers have gone over the expressive advantages of syntax-based approaches. But it is rare to see an in-depth, quantitative study of strengths and weaknesses of particular models with respect to each other. This is important for a scientific understanding of how these models work in practice. Our main novel contribution is a comparison of phrase-based and syntax-based extraction methods and phrase pair coverage. We also add to the literature a new method of improving that coverage. Additionally, we do a careful study of several syntax-based extraction techniques, testing whether (and how much) they affect phrase pair coverage, and whether (and how much) they affect end-to-end MT accuracy. The MT accuracy tests are needed because we want to see the individual effects of particular techniques under the same testing conditions. For this comparison, we choose a previously established statistical phrase-based model (Och and Ney, 2004) and a previously established statistical string-to-tree model (Galley et al., 2004). These two models are chosen because they are the basis of two of the most successful systems in the NIST 2006 MT

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 755–763, Prague, June 2007. ©2007 Association for Computational Linguistics

evaluation[1].

2 Phrase-based Extraction

The Alignment Template system (ATS) described by Och and Ney (2004) is representative of statistical phrase-based models. The basic unit of translation is the phrase pair, which consists of a sequence of words in the source language, a sequence of words in the target language, and a vector of feature values which describe this pair's likelihood. Decoding produces a string in the target language, in order, from beginning to end. During decoding, features from each phrase pair are combined with other features (e.g., re-ordering, language models) using a log-linear model to compute the score of the entire translation.

The ATS phrase extraction algorithm learns these phrase pairs from an aligned, parallel corpus. This corpus is conceptually a list of tuples of <source sentence, target sentence, bi-directional word alignments> which serve as training examples, one of which is shown in Figure 1.

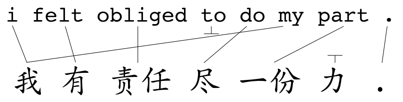


Figure 1: a phrase-based training example

For each training example, the algorithm identifies and extracts all pairs of <source sequence, target sequence> that are consistent with the alignments. It does this by first enumerating all source-side word sequences up to a length limit L, and for each source sequence, it identifies all target words aligned to those source words. For example, in Figure 1, for the source phrase 有 责任 尽, the target words it aligns to are `felt`, `obliged`, and `do`. These words, and all those between them, are the proposed target phrase. If no words in the proposed target phrase align to words outside of the source phrase, then this phrase pair is extracted.

The extraction algorithm can also look to the left and right of the proposed target phrase for neighboring unaligned words and extracts phrases. For example, for the phrase pair 有 责任 ↔ `felt obliged`, the word `to` is a neighboring unaligned word. It constructs new target phrases by adding on consecutive unaligned words in both directions, and extracts those in new pairs, too (e.g., 有 责任 ↔ `felt obliged to`). For efficiency reasons, implementations often skip this step.

Figure 2 shows the complete set of phrase pairs up to length 4 that are extracted from the Figure 1 training example. Notice that no extracted phrase pair contains the character 我. Because of the alignments, the smallest legal phrase pair, 我 有 责任 尽 ↔ `i felt obliged to do my`, is beyond the size limit of 4, so it is not extracted in this example.

有	↔	`felt`
有 责任	↔	`felt obliged`
有 责任 尽	↔	`felt obliged to do`
责任	↔	`obliged`
责任 尽	↔	`obliged to do`
尽	↔	`do`
一份	↔	`part`
一份 力	↔	`part`
一份 力 .	↔	`part .`
力 .	↔	`.`
.	↔	`.`

Figure 2: phrases up to length 4 extracted from the example in Figure 1

Phrase pairs are extracted over the entire training corpus. Due to differing alignments, some phrase pairs that cannot be learned from one example may be learned from another. These pairs are then counted, once for each time they are seen in a training example, and these counts are used as the basis for maximum likelihood probability features, such as $p(f|e)$ and $p(e|f)$.

3 Syntax-based Extraction

The GHKM syntax-based extraction method for learning statistical syntax-based translation rules, presented first in (Galley et al., 2004) and expanded on in (Galley et al., 2006), is similar to phrase-based extraction in that it extracts rules consistent with given word alignments. A primary difference is the use of syntax trees on the target side, rather than sequences of words. The basic unit of translation is the translation rule, consisting of a sequence of words

[1] `http://www.nist.gov/speech/tests/mt/mt06eval_official_results.html`

and variables in the source language, a syntax tree in the target language having words or variables at the leaves, and again a vector of feature values which describe this pair's likelihood. Translation rules can:

- look like phrase pairs with syntax decoration:

  NPB(NNP(prime)
  NNP(minister)
  NNP(keizo)
  NNP(obuchi)) ↔ 小渊惠三 首相

- carry extra contextual constraints:

  VP(VBD(said)
  x_0:SBAR-C) ↔ 说 x_0

  (according to this rule, 说 can translate to said only if some Chinese sequence to the right of 说 is translated into an SBAR-C)

- be non-constituent phrases:

  VP(VBD(said)
  SBAR-C(IN(that)
  x_0:S-C)) ↔ 说 x_0

  VP(VBD(pointed)
  PRT(RP(out))
  x_0:SBAR-C) ↔ 指出 x_0

- contain non-contiguous phrases, effectively "phrases with holes":

  PP(IN(on)
  NP-C(NPB(DT(the)
  x_0:NNP))
  NN(issue)))) ↔ 在 x_0 问题 上

  PP(IN(on)
  NP-C(NPB(DT(the)
  NN(issue))
  x_0:PP)) ↔ 在 x_0 问题 上

- be purely structural (no words):

  S(x_0:NP-C x_1:VP) ↔ x_0 x_1

- re-order their children:

  NP-C(NPB(DT(the)
  x_0:NN)
  PP(IN(of)
  x_1:NP-C)) ↔ x_1 的 x_0

Decoding with this model produces a tree in the target language, bottom-up, by parsing the foreign string using a CYK parser and a binarized rule set (Zhang et al., 2006). During decoding, features from each translation rule are combined with a language model using a log-linear model to compute the score of the entire translation.

The GHKM extractor learns translation rules from an aligned parallel corpus where the target side has been parsed. This corpus is conceptually a list of tuples of <source sentence, target tree, bi-directional word alignments> which serve as training examples, one of which is shown in Figure 3.

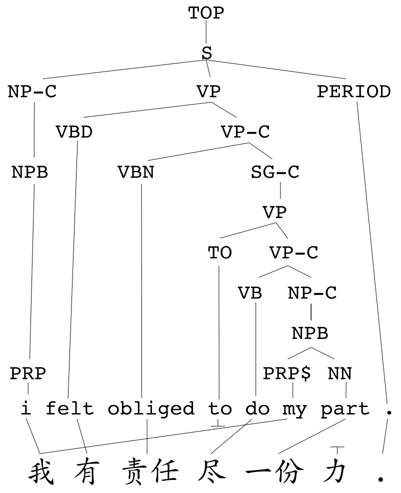


Figure 3: a syntax-based training example

For each training example, the GHKM extractor computes the set of minimally-sized translation rules that can explain the training example while remaining consistent with the alignments. This is, in effect, a non-overlapping tiling of translation rules over the tree-string pair. If there are no unaligned words in the source sentence, this is a unique set. This set, ordered into a tree of rule applications, is called the derivation tree of the training example. Unlike the ATS model, there are no inherent size limits, just the constraint that the rules be as small as possible for the example.

Ignoring the unaligned 力 for the moment, there are seven minimal translation rules that are extracted from the example in Figure 3, as shown in Figure 4. Notice that rule 6 is rather large and applies to a very limited syntactic context. The only constituent node that covers both i and my is the S, so the rule rooted at S is extracted, with variables for every branch below this top constituent that can be explained by other rules. Note also that to becomes a part of this rule naturally. If the alignments were not as constraining (e.g., if my was unaligned), then instead of this one big rule many smaller rules would be extracted, such as structural rules (e.g., VP(x_0:VBD x_1:VP-C) ↔ x_0 x_1) and function word insertion rules (e.g., VP(TO(to) x_0:VP-C) ↔ x_0).

1. VBD(felt) ↔ 有
2. VBN(obliged) ↔ 责任
3. VB(do) ↔ 尽
4. NN(part) ↔ 一份
5. PERIOD(.) ↔ .
6. S(NP-C(NPB(PRP(I)))
   VP(x_0:VBD
   VP-C(x_1:VBN
   SG-C(VP(TO(to)
   VP-C(x_2:VB
   NP-C(NPB(PRP\$(my)
   x_3:NN)))))))
   x_4:PERIOD) ↔ 我 x_0 x_1 x_2 x_3 x_4
7. TOP(x_0:S) ↔ x_0

Figure 4: rules extracted from training example

We ignored unaligned source words in the example above. Galley et al. (2004) attach the unaligned source word to the highest possible location, in our example, the S. Thus it is extracted along with our large rule 6, changing the target language sequence to "我 x_0 x_1 x_2 x_3 力 x_4". This treatment still results in a unique derivation tree no matter how many unaligned words are present.

In Galley et al. (2006), instead of a unique derivation tree, the extractor computes several derivation trees, each with the unaligned word added to a different rule such that the data is still explained. For example, for the tree-string pair in Figure 3, 力 could be added not only to rule 6, but alternatively to rule 4 or 5, to make the new rules:

NN(part) ↔ 一份 力
PERIOD(.) ↔ 力 .

This results in three different derivations, one with the 力 character in rule 4 (with rules 5 and 6 as originally shown), another with the 力 character in rule 5 (with rules 4 and 6 as originally shown), and lastly one with the 力 character in rule 6 (with rules 4 and 5 as originally shown) as in the original paper (Galley et al., 2004). In total, ten different rules are extracted from this training example.

As with ATS, translation rules are extracted and counted over the entire training corpus, a count of one for each time they appear in a training example. These counts are used to estimate several features, including maximum likelihood probability features for $p(e_{tree}, f_{words}|e_{head})$, $p(e_{words}|f_{words})$, and $p(f_{words}|e_{words})$.

4 Differences in Phrasal Coverage

Both the ATS model and the GHKM model extract linguistic knowledge from parallel corpora, but each has fundamentally different constraints and assumptions. To compare the models empirically, we extracted phrase pairs (for the ATS model) and translation rules (for the GHKM model) from parallel training corpora described in Table 1. The ATS model was limited to phrases of length 10 on the source side, and length 20 on the target side. A superset of the parallel data was word aligned by GIZA union (Och and Ney, 2003) and EMD (Fraser and Marcu, 2006). The English side of training data was parsed using an implementation of Collins' model 2 (Collins, 2003).

	Chinese	Arabic
Document IDs	LDC2003E07	LDC2004T17
	LDC2003E14	LDC2004T18
	LDC2005T06	LDC2005E46
# of segments	329,031	140,511
# of words in foreign corpus	7,520,779	3,147,420
# of words in English corpus	9,864,294	4,067,454

Table 1: parallel corpora used to train both models

Table 2 shows the total number of GHKM rules extracted, and a breakdown of the different kinds of rules. Non-lexical rules are those whose source side is composed entirely of variables — there are no source words in them. Because of this, they potentially apply to any sentence. Lexical rules (their counterpart) far outnumber non-lexical rules. Of the lexical rules, a rule is considered a *phrasal rule* if its source side and the yield of its target side contain exactly one contiguous phrase each, optionally with one or more variables on either side of the phrase. Non-phrasal rules include structural rules, re-ordering rules, and non-contiguous phrases. These rules are not easy to directly compare to any phrase pairs from the ATS model, so we do not focus on them here.

Phrasal rules can be directly compared to ATS phrase pairs, the easiest way being to discard the

Statistic	Chinese	Arabic
total translation rules	2,487,110	662,037
non-lexical rules	110,066	15,812
lexical rules	2,377,044	646,225
phrasal rules	1,069,233	406,020
distinct GHKM-derived phrase pairs	919,234	352,783
distinct corpus-specific GHKM-derived phrase pairs	203,809	75,807

Table 2: a breakdown of how many rules the GHKM extraction algorithm produces, and how many phrase pairs can be derived from them

syntactic context and look at the phrases contained in the rules. The second to last line of Table 2 shows the number of phrase pairs that can be derived from the above phrasal rules. The number of GHKM-derived phrase pairs is lower than the number of phrasal rules because some rules represent the same phrasal translation, but with different syntactic contexts. The last line of Table 2 shows the subset of phrase pairs that contain source phrases found in our development corpus.

Table 3 compares these corpus-specific GHKM-derived phrase pairs with the corpus-specific ATS phrase pairs. Note that the number of phrase pairs derived from the GHKM rules is less than the number of phrase pairs extracted by ATS. Moreover, only slightly over half of the phrase pairs extracted by the ATS model are common to both models. The limits and constraints of each model are responsible for this difference in contiguous phrases learned.

Source of phrase pairs	Chinese	Arabic
GHKM-derived	203,809	75,807
ATS	295,537	133,576
Overlap between models	160,901	75,038
GHKM only	42,908	769
ATS only	134,636	58,538
ATS-useful only	1,994	2,199

Table 3: comparison of corpus-specific phrase pairs from each model

GHKM learns some contiguous phrase pairs that the phrase-based extractor does not. Only a small portion of these are due to the fact that the GHKM model has no inherent size limit, while the phrase based system has limits. More numerous are cases where unaligned English words are not added to an ATS phrase pair while GHKM adopts them at a syntactically motivated location, or where a larger rule contains mostly syntactic structure but happens to have some unaligned words in it. For example, consider Figure 5. Because `basic` and `will` are unaligned, ATS will learn no phrase pairs that translate to these words alone, though they will be learned as a part of larger phrases.

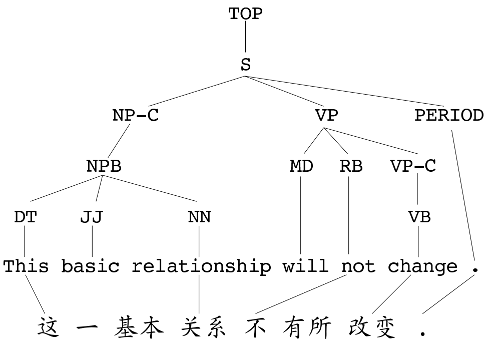

TOP
S
NP-C
VP
PERIOD
NPB
MD
RB
VP-C
DT
JJ
NN
VB
This basic relationship will not change .
这 一 基本 关系 不 有所 改变 .

Figure 5: Situation where GHKM is able to learn rules that translate into `basic` and `will`, but ATS is not

GHKM, however, will learn several phrasal rules that translate to `basic`, based on the syntactic context

NPB(x_0:DT
JJ(basic) $\leftrightarrow x_0$ 一 x_1
x_1:NN)

NPB(x_0:DT
JJ(basic) $\leftrightarrow x_0$ 一 基本 x_1
x_1:NN)

NPB(x_0:DT
JJ(basic) $\leftrightarrow x_0$ 基本 x_1
x_1:NN)

and one phrasal rule that translates into `will`

VP(MD(will)
x_0:RB $\leftrightarrow x_0$ 有所 x_1
x_1:VP-C)

The quality of such phrases may vary. For example, the first translation of 一 (literally: “one” or “a”) to `basic` above is a phrase pair of poor quality, while the other two for `basic` and one for `will` are arguably reasonable.

However, Table 3 shows that ATS was able to learn many more phrase pairs that GHKM was not. Even more significant is the subset of these missing phrase pairs that the ATS decoder used in its best[2]

[2] i.e. highest scoring

translation of the corpus. According to the phrase-based system these are the most “useful” phrase pairs and GHKM could not learn them. Since this is a clear deficiency, we will focus on analyzing these phrase pairs (which we call *ATS-useful*) and the reasons they were not learned.

Table 4 shows a breakdown, categorizing each of these missing ATS-useful phrase pairs and the reasons they were not able to be learned. The most common reason is straightforward: by extracting only the minimally-sized rules, GHKM is unable to learn many larger phrases that ATS learns. If GHKM can make a word-level analysis, it will do that, at the expense of a phrase-level analysis. Galley et al. (2006) propose one solution to this problem and Marcu et al. (2006) propose another, both of which we explore in Sections 5.1 and 5.2.

Category of missing ATS-useful phrase pairs	Chinese	Arabic
Not minimal	1,320	1,366
Extra target words in GHKM rules	220	27
Extra source words in GHKM rules	446	799
Other (e.g. parse failures)	8	7
Total missing useful phrase pairs	1,994	2,199

Table 4: reasons that ATS-useful phrase pairs could not be extracted by GHKM as phrasal rules

The second reason is that the GHKM model is sometimes forced by its syntactic constraints to include extra words. Sometimes this is only target language words, and this is often useful — the rules are learning to insert these words in their proper context. But most of the time, source language words are also forced to be part of the rule, and this is harmful — it makes the rules less general. This latter case is often due to poorly aligned target language words (such as the 我 in our Section 3 rule extraction example), or unaligned words under large, flat constituents.

Another factor here: some of the phrase pairs are learned by both systems, but GHKM is more specific about the context of use. This can be both a strength and a weakness. It is a strength when the syntactic context helps the phrase to be used in a syntactically correct way, as in

$$\begin{array}{l}\texttt{VP(VBD(said)}\\ \quad x_0\texttt{:SBAR-C)}\end{array} \leftrightarrow \text{说}\; x_0$$

where the syntax rule requires a constituent of type SBAR-C. Conversely its weakness is seen when the context is too constrained. For example, ATS can easily learn the phrase

$$\text{总理} \leftrightarrow \texttt{prime minister}$$

and is then free to use it in many contexts. But GHKM learns 45 different rules, each that translate this phrase pair in a unique context. Figure 6 shows a sampling. Notice that though many variations are present, the decoder is unable to use any of these rules to produce certain noun phrases, such as “current Japanese Prime Minister Shinzo Abe”, because no rule has the proper number of English modifiers.

NPB(NNP(prime) NNP(minister) x_0:NNP) ↔ x_0 总理
NPB(x_0:NNP NNP(prime) NNP(minister) x_1:NNP) ↔ x_0 总理 x_1
NPB(x_0:JJ NNP(prime) NNP(minister) x_1:NNP) ↔ x_0 总理 x_1
NPB(NNP(prime) NNP(minister) x_0:NNP) ↔ 总理 x_0
NPB(NNP(prime) NNP(minister)) ↔ 总理
NPB(NNP(prime) NNP(minister) x_0:NNP x_1:NNP) ↔ x_0 x_1 总理
NPB(x_0:DT x_1:JJ JJ(prime) NN(minister)) ↔ x_0 x_1 总理
NPB(x_0:NNP NNP(prime) NNP(minister) x_1:NNP) ↔ x_0 总理 x_1
NPB(x_0:NNP NNP(prime) NNP(minister) x_1:NNP) ↔ x_0 总理 x_1

Figure 6: a sampling of the 45 rules that translate 总理 to prime minister

5 Coverage Improvements

Each of the models presented so far has advantages and disadvantages. In this section, we consider ideas that make up for deficiencies in the GHKM model, drawing our inspiration from the strong points of the ATS model. We then measure the effects of each idea empirically, showing both what is gained and the potential limits of each modification.

5.1 Composed Rules

Galley et al. (2006) proposed the idea of composed rules. This removes the minimality constraint required earlier: any two or more rules in a parent-child relationship in the derivation tree can be combined to form a larger, composed rule. This change is similar in spirit to the move from word-based to phrase-based MT models, or parsing with a DOP model (Bod et al., 2003) rather than a plain PCFG.

Because this results in exponential variations, a size limit is employed: for any two or more rules to be allowed to combine, the size of the resulting rule must be at most n. The size of a rule is defined as the number of non-part-of-speech, non-leaf constituent labels in a rule's target tree. For example, rules 1-5 shown in Section 3 have a size of 0, and rule 6 has a size of 10. Composed rules are extracted in addition to minimal rules, which means that a larger n limit always results in a superset of the rules extracted when a smaller n value is used. When n is set to 0, then only minimal rules are extracted. Table 5 shows the growth in the number of rules extracted for several size limits.

Size limit (n)	Chinese	Arabic
0 (minimal)	2,487,110	662,037
2	12,351,297	2,742,513
3	26,917,088	4,824,928
4	55,781,061	8,487,656

Table 5: increasing the size limit of composed rules significantly increases the number of rules extracted

In our previous analysis, the main reason that GHKM did not learn translations for ATS-useful phrase pairs was due to its minimal-only approach. Table 6 shows the effect that composed rule extraction has on the total number of ATS-useful phrases missing. Note that as the allowed size of composed rule increases, we are able to extract an greater percentage of the missing ATS-useful phrase pairs.

Size limit (n)	Chinese	Arabic
0 (minimal)	1,994	2,199
2	1,478	1,528
3	1,096	1,210
4	900	1,041

Table 6: number of ATS-useful phrases still missing when using GHKM composed rule extraction

Unfortunately, a comparison of Tables 5 and 6 indicates that the number of ATS-useful phrase pairs gained is growing at a much slower rate than the total number of rules. From a practical standpoint, more rules means more processing work and longer decoding times, so there are diminishing returns from continuing to explore larger size limits.

5.2 SPMT Model 1 Rules

An alternative for extracting larger rules called SPMT model 1 is presented by Marcu et al. (2006). Though originally presented as a separate model, the method of rule extraction itself builds upon the minimal GHKM method just as composed rules do. For each training example, the method considers all source language phrases up to length L. For each of these phrases, it extracts the smallest possible syntax rule that does not violate the alignments. Table 7 shows that this method is able to extract rules that cover useful phrases, and can be combined with size 4 composed rules to an even better effect. Since there is some overlap in these methods, when combining the two methods we eliminate any redundant rules.

Method	Chinese	Arabic
composed alone (size 4)	900	1,041
SPMT model 1 alone	676	854
composed + SPMT model 1	663	835

Table 7: ATS-useful phrases still missing after different non-minimal methods are applied

Note that having more phrasal rules is not the only advantage of composed rules. Here, combining both composed and SPMT model 1 rules, our gain in useful phrases is not very large, but we do gain additional, larger syntax rules. As discussed in (Galley et al., 2006), composed rules also allow the learning of more context, such as

ADJP(ADVP(RB(far)
CC(and)
RB(away) ↔ 远远 x_0
x_0:JJ)

This rule is not learned by SPMT model 1 because it is not the smallest rule that can explain the phrase pair, but it is still valuable for its syntactic context.

5.3 Restructuring Trees

Table 8 updates the causes of missing ATS-useful phrase pairs. Most are now caused by syntactic constraints, thus we need to address these in some way.

GHKM translation rules are affected by large, flat constituents in syntax trees, as in the `prime minister` example earlier. One way to soften this constraint is to binarize the trees, so that wide constituents are broken down into multiple levels of tree structure. The approach we take here is head-out binarization (Wang et al., 2007), where any constituent with more than two children is split into partial constituents. The children to the left of the head word

Category of ATS-useful phrase pairs	Chinese	Arabic
Too large	12	9
Extra target words in GHKM rules	218	27
Extra source words in GHKM rules	424	792
Other (e.g. parse failures)	9	7
Total missing useful phrase pairs	663	835

Table 8: reasons that ATS-useful phrase pairs are still not extracted as phrasal rules, with composed and SPMT model 1 rules in place

are binarized one direction, while the children to the right are binarized the other direction. The top node retains its original label (e.g. NPB), while the new partial constituents are labeled with a bar (e.g. $\overline{\text{NPB}}$). Figure 7 shows an example.

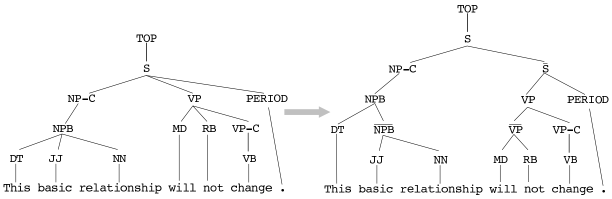

TOP
S
NP-C
VP
PERIOD
NPB
MD
RB
VP-C
DT
JJ
NN
VB
This basic relationship will not change .
TOP
S
NP-C
S̄
NPB
VP
PERIOD
DT
NPB̄
VP̄
VP-C
JJ
NN
MD
RB
VB
This basic relationship will not change .

Figure 7: head-out binarization in the target language: S, NPB, and VP are binarized according to the head word

Table 9 shows the effect of binarization on phrasal coverage, using both composed and SPMT rules. By eliminating some of the syntactic constraints we allow more freedom, which allows increased phrasal coverage, but generates more rules.

Category of missing ATS-useful phrase pairs	Chinese	Arabic
Too large	16	12
Extra target words in GHKM rules	123	12
Extra source words in GHKM rules	307	591
Other (e.g. parse failures)	12	7
Total missing useful phrase pairs	458	622

Table 9: reasons that ATS-useful phrase pairs still could not be extracted as phrasal rules after binarization

6 Evaluation of Translations

To evaluate translation quality of each of these models and methods, we ran the ATS decoder using its extracted phrase pairs and the syntax-based decoder using all the rule sets mentioned above. Table 10 describes the development and test datasets used, along with four references for measuring BLEU. Tuning was done using Maximum BLEU hill-climbing (Och, 2003). Features used for the ATS system were the standard set. For the syntax-based translation system, we used a similar set of features.

		# of lines	
	Dataset	Chinese	Arabic
Development set	NIST 2002 MT eval (sentences $<$ 47 tokens)	925	696
Test set	NIST 2003 MT eval	919	663

Table 10: development and test corpora

Table 11 shows the case-insensitive NIST BLEU4 scores for both our development and test decodings. The BLEU scores indicate, first of all, that the syntax-based system is much stronger in translating Chinese than Arabic, in comparison to the phrase-based system. Also, the ideas presented here for improving phrasal coverage generally improve the syntax-based translation quality. In addition, composed rules are shown to be helpful as compared to the minimal runs. This is true even when SPMT model 1 is added, which indicates that the size 4 composed rules bring more than just improved phrasal coverage.

	Chinese		Arabic	
Experiment	Dev	Test	Dev	Test
Baseline ATS	34.94	32.83	50.46	50.52
Baseline GHKM (minimal only)	38.02	37.67	49.34	49.99
GHKM composed size 2	40.24	39.75	50.76	50.94
GHKM composed size 3	40.95	40.44	51.56	51.48
GHKM composed size 4	41.36	40.69	51.60	51.71
GHKM minimal + SPMT model 1	39.78	39.16	50.17	51.27
GHKM composed + SPMT model 1	42.04	41.07	51.73	51.53
With binarization	42.17	41.26	52.50	51.79

Table 11: evaluation results (reported in case-insensitive NIST BLEU4)

7 Conclusions

Both the ATS model for phrase-based machine translation and the GHKM model for syntax-based machine translation are state-of-the-art methods. Each extraction method has strengths and weaknesses as compared to the other, and there are surprising differences in phrasal coverage — neither is merely a superset of the other. We have shown that it is possible to gain insights from the strengths of the phrase-based extraction model to increase both the phrasal coverage and translation accuracy of the syntax-based model.

However, there is still room for improvement in both models. For syntax models, there are still holes in phrasal coverage, and other areas are needing progress, such as decoding efficiency. For phrase-based models, incorporating syntactic knowledge and constraints may lead to improvements as well.

8 Acknowledgments

The authors wish to acknowledge our colleagues at ISI, especially David Chiang, for constructive criticism on an early draft of this document, and several reviewers for their detailed comments which helped us make the paper stronger. We are also grateful to Jens-Sönke Vöckler for his assistance in setting up an experimental pipeline, without which this work would have been much more tedious and difficult. This research was supported under DARPA Contract No. HR0011-06-C-0022.

References

Rens Bod, Remko Scha, and Khalil Sima'an, editors. 2003. *Data-Oriented Parsing*. CSLI Publications, University of Chicago Press.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL 2005*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4).

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proc. EMNLP 2006*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. ACL 2003*.

Alexander Fraser and Daniel Marcu. 2006. Semi-supervised training for statistical word alignment. In *Proc. ACL 2006*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL 2004*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL 2006*.

Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proc. ACL 2003, companion volume*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA 2006*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL 2003*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. EMNLP 2006*.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proc. ACL 2004*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL 2003*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. ACL 2005*.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. EMNLP and CoNLL 2007*.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proc. ACL 1998*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. ACL 2001*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. NAACL HLT 2006*.

Online Large-Margin Training for Statistical Machine Translation

Taro Watanabe   Jun Suzuki   Hajime Tsukada   Hideki Isozaki
NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan
{taro,jun,tsukada,isozaki}@cslab.kecl.ntt.co.jp

Abstract

We achieved a state of the art performance in statistical machine translation by using a large number of features with an online large-margin training algorithm. The millions of parameters were tuned only on a small development set consisting of less than 1K sentences. Experiments on Arabic-to-English translation indicated that a model trained with sparse binary features outperformed a conventional SMT system with a small number of features.

1 Introduction

The recent advances in statistical machine translation have been achieved by discriminatively training a small number of real-valued features based either on (hierarchical) phrase-based translation (Och and Ney, 2004; Koehn et al., 2003; Chiang, 2005) or syntax-based translation (Galley et al., 2006). However, it does not scale well with a large number of features of the order of millions.

Tillmann and Zhang (2006), Liang et al. (2006) and Bangalore et al. (2006) introduced sparse binary features for statistical machine translation trained on a large training corpus. In this framework, the problem of translation is regarded as a sequential labeling problem, in the same way as part-of-speech tagging, chunking or shallow parsing. However, the use of a large number of features did not provide any significant improvements over a conventional small feature set.

Bangalore et al. (2006) trained the lexical choice model by using Conditional Random Fields (CRF) realized on a WFST. Their modeling was reduced to Maximum Entropy Markov Model (MEMM) to handle a large number of features which, in turn, faced the labeling bias problem (Lafferty et al., 2001). Tillmann and Zhang (2006) trained their feature set using an online discriminative algorithm. Since the decoding is still expensive, their online training approach is approximated by enlarging a merged *k*-best list one-by-one with a 1-best output. Liang et al. (2006) introduced an averaged perceptron algorithm, but employed only 1-best translation. In Watanabe et al. (2006a), binary features were trained only on a small development set using a variant of voted perceptron for reranking *k*-best translations. Thus, the improvement is merely relative to the baseline translation system, namely whether or not there is a good translation in their *k*-best.

We present a method to estimate a large number of parameters — of the order of millions — using an online training algorithm. Although it was intuitively considered to be prone to overfitting, training on a small development set — less than 1K sentences — was sufficient to achieve improved performance. In this method, each training sentence is decoded and weights are updated at every iteration (Liang et al., 2006). When updating model parameters, we employ a memorization-variant of a local updating strategy (Liang et al., 2006) in which parameters are optimized toward a set of good translations found in the *k*-best list across iterations. The objective function is an approximated BLEU (Watanabe et al., 2006a) that scales the loss of a sentence BLEU to a document-wise loss. The parameters are trained using the

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 764–773, Prague, June 2007. ©2007 Association for Computational Linguistics

Margin Infused Relaxed Algorithm (MIRA) (Crammer et al., 2006). MIRA is successfully employed in dependency parsing (McDonald et al., 2005) or the joint-labeling/chunking task (Shimizu and Haas, 2006). Experiments were carried out on an Arabic-to-English translation task, and we achieved significant improvements over conventional minimum error training with a small number of features.

This paper is organized as follows: First, Section 2 introduces the framework of statistical machine translation. As a baseline SMT system, we use the hierarchical phrase-based translation with an efficient left-to-right generation (Watanabe et al., 2006b) originally proposed by Chiang (2005). In Section 3, a set of binary sparse features are defined including numeric features for our baseline system. Section 4 introduces an online large-margin training algorithm using MIRA with our key components. The experiments are presented in Section 5 followed by discussion in Section 6.

2 Statistical Machine Translation

We use a log-linear approach (Och, 2003) in which a foreign language sentence f is translated into another language, for example English, e, by seeking a maximum solution:

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, \mathbf{w}^T \cdot \mathbf{h}(f, e) \tag{1}$$

where $\mathbf{h}(f, e)$ is a large-dimension feature vector. $\mathbf{w}$ is a weight vector that scales the contribution from each feature. Each feature can take any real value, such as the log of the n-gram language model to represent fluency, or a lexicon model to capture the word or phrase-wise correspondence.

2.1 Hierarchical Phrase-based SMT

Chiang (2005) introduced the hierarchical phrase-based translation approach, in which non-terminals are embedded in each phrase. A translation is generated by hierarchically combining phrases using the non-terminals. Such a quasi-syntactic structure can naturally capture the reordering of phrases that is not directly modeled by a conventional phrase-based approach (Koehn et al., 2003). The non-terminal embedded phrases are learned from a bilingual corpus without a linguistically motivated syntactic structure.

Based on hierarchical phrase-based modeling, we adopted the left-to-right target generation method (Watanabe et al., 2006b). This method is able to generate translations efficiently, first, by simplifying the grammar so that the target side takes a phrase-prefixed form, namely a target normalized form. Second, a translation is generated in a left-to-right manner, similar to the phrase-based approach using Earley-style top-down parsing on the source side. Coupled with the target normalized form, n-gram language models are efficiently integrated during the search even with a higher order of n.

2.2 Target Normalized Form

In Chiang (2005), each production rule is restricted to a rank-2 or binarized form in which each rule contains at most two non-terminals. The target normalized form (Watanabe et al., 2006b) further imposes a constraint whereby the target side of the aligned right-hand side is restricted to a Greibach Normal Form like structure:

$$X \rightarrow \left\langle \gamma, \bar{b}\beta, \sim \right\rangle \tag{2}$$

where X is a non-terminal, γ is a source side string of arbitrary terminals and/or non-terminals. $\bar{b}\beta$ is a corresponding target side where $\bar{b}$ is a string of terminals, or a phrase, and β is a (possibly empty) string of non-terminals. $\sim$ defines one-to-one mapping between non-terminals in γ and β. The use of phrase $\bar{b}$ as a prefix maintains the strength of the phrase-base framework. A contiguous English side with a (possibly) discontiguous foreign language side preserves phrase-bounded local word reordering. At the same time, the target normalized framework still combines phrases hierarchically in a restricted manner.

2.3 Left-to-Right Target Generation

Decoding is performed by parsing on the source side and by combining the projected target side. We applied an Earley-style top-down parsing approach (Wu and Wong, 1998; Watanabe et al., 2006b; Zollmann and Venugopal, 2006). The basic idea is to perform top-down parsing so that the projected target side is generated in a left-to-right manner. The search is guided with a push-down automaton, which keeps track of the span of uncovered source word positions. Combined with the rest-cost estimation aggregated in a bottom-up way, our decoder efficiently searches for the most likely translation.

The use of a target normalized form further simplifies the decoding procedure. Since the rule form does not allow any holes for the target side, the integration with an *n*-gram language model is straightforward: the prefixed phrases are simply concatenated and intersected with *n*-gram.

3 Features

3.1 Baseline Features

The hierarchical phrase-based translation system employs standard numeric value features:

- *n*-gram language model to capture the fluency of the target side.
- Hierarchical phrase translation probabilities in both directions, $h(\gamma|\bar{b}\beta)$ and $h(\bar{b}\beta|\gamma)$, estimated by relative counts, count$(\gamma, \bar{b}\beta)$.
- Word-based lexically weighted models of $h_{lex}(\gamma|\bar{b}\beta)$ and $h_{lex}(\bar{b}\beta|\gamma)$ using lexical translation models.
- Word-based insertion/deletion penalties that penalize through the low probabilities of the lexical translation models (Bender et al., 2004).
- Word/hierarchical-phrase length penalties.
- Backtrack-based penalties inspired by the distortion penalties in phrase-based modeling (Watanabe et al., 2006b).

3.2 Sparse Features

In addition to the baseline features, a large number of binary features are integrated in our MT system. We may use any binary features, such as

$$h(f, e) = \begin{cases} 1 & \text{English word “violate” and Arabic} \\ & \text{word “tnthk” appeared in } e \text{ and } f. \\ 0 & \text{otherwise.} \end{cases}$$

The features are designed by considering the decoding efficiency and are based on the word alignment structure preserved in hierarchical phrase translation pairs (Zens and Ney, 2006). When hierarchical phrases are extracted, the word alignment is preserved. If multiple word alignments are observed with the same source and target sides, only the frequently observed word alignment is kept to reduce the grammar size.

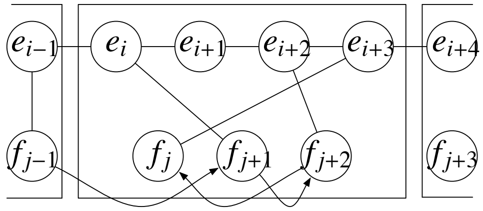


Figure 1: An example of sparse features for a phrase translation.

3.2.1 Word Pair Features

Word pair features reflect the word correspondence in a hierarchical phrase. Figure 1 illustrates an example of sparse features for a phrase translation pair $f_j, ..., f_{j+2}$ and $e_i, ..., e_{i+3}$ [1]. From the word alignment encoded in this phrase, we can extract word pair features of (e_i, f_{j+1}), (e_{i+2}, f_{j+2}) and (e_{i+3}, f_j).

The bigrams of word pairs are also used to capture the contextual dependency. We assume that the word pairs follow the target side ordering. For instance, we define $((e_{i-1}, f_{j-1}), (e_i, f_{j+1}))$, $((e_i, f_{j+1}), (e_{i+2}, f_{j+2}))$ and $((e_{i+2}, f_{j+2}), (e_{i+3}, f_j))$ indicated by the arrows in Figure 1.

Extracting bigram word pair features following the target side ordering implies that the corresponding source side is reordered according to the target side. The reordering of hierarchical phrases is represented by using contextually dependent word pairs across their boundaries, as with the feature $((e_{i-1}, f_{j-1}), (e_i, f_{j+1}))$ in Figure 1.

3.2.2 Insertion Features

The above features are insufficient to capture the translation because spurious words are sometimes inserted in the target side. Therefore, insertion features are integrated in which no word alignment is associated in the target. The inserted words are associated with all the words in the source sentence, such as $(e_{i+1}, f_1), ..., (e_{i+1}, f_J)$ for the non-aligned word e_{i+1} with the source sentence f_1^J in Figure 1. In the

[1] For simplicity, we show an example of phrase translation pairs, but it is trivial to define the features over hierarchical phrases.

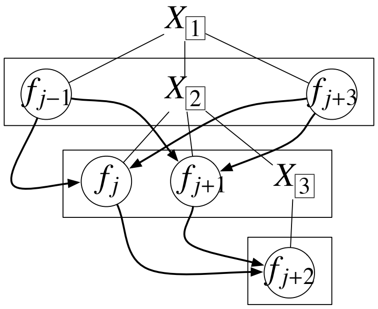


Figure 2: Example hierarchical features.

same way, we will be able to include deletion features where a non-aligned source word is associated with the target sentence. However, this would lead to complex decoding in which all the translated words are memorized for each hypothesis, and thus not integrated in our feature set.

3.2.3 Target Bigram Features

Target side bigram features are also included to directly capture the fluency as in the *n*-gram language model (Roark et al., 2004). For instance, bigram features of $(e_{i-1}, e_i), (e_i, e_{i+1}), (e_{i+1}, e_{i+2})$... are observed in Figure 1.

3.2.4 Hierarchical Features

In addition to the phrase motivated features, we included features inspired by the hierarchical structure. Figure 2 shows an example of hierarchical phrases in the source side, consisting of $X_{\boxed{1}} \rightarrow \left\langle f_{j-1} X_{\boxed{2}} f_{j+3} \right\rangle$, $X_{\boxed{2}} \rightarrow \left\langle f_j f_{j+1} X_{\boxed{3}} \right\rangle$ and $X_{\boxed{3}} \rightarrow \left\langle f_{j+2} \right\rangle$. Hierarchical features capture the dependency of the source words in a parent phrase to the source words in child phrases, such as (f_{j-1}, f_j), (f_{j-1}, f_{j+1}), (f_{j+3}, f_j), (f_{j+3}, f_{j+1}), (f_j, f_{j+2}) and (f_{j+1}, f_{j+2}) as indicated by the arrows in Figure 2. The hierarchical features are extracted only for those source words that are aligned with the target side to limit the feature size.

3.3 Normalization

In order to achieve the generalization capability, the following normalized tokens are introduced for each surface form:

- Word class or POS.

- 4-letter prefix and suffix. For instance, the word “violate” is normalized to “viol+” and “+late” by taking the prefix and suffix, respectively.

- Digits replaced by a sequence of “@”. For example, the word “2007/6/27” is represented as “@@@@/@/@@”.

Algorithm 1 Online Training Algorithm

```
   Training data: $\mathcal{T} = \{(f^t, \mathbf{e}^t)\}_{t=1}^{T}$
   $m$-best oracles: $\mathcal{O} = \{\}_{t=1}^{T}$
   $i = 0$
1: for $n = 1, ..., N$ do
2:     for $t = 1, ..., T$ do
3:         $\mathcal{C}^t \leftarrow \text{best}_k(f^t; \mathbf{w}^i)$
4:         $\mathcal{O}^t \leftarrow \text{oracle}_m(\mathcal{O}^t \cup \mathcal{C}^t; \mathbf{e}^t)$
5:         $\mathbf{w}^{i+1}$ = update $\mathbf{w}^i$ using $\mathcal{C}^t$ w.r.t. $\mathcal{O}^t$
6:         $i = i + 1$
7:     end for
8: end for
9: return $\frac{\sum_{i=1}^{NT} \mathbf{w}^i}{NT}$
```

We consider all possible combination of those token types. For example, the word pair feature (violate, tnthk) is normalized and expanded to (viol+, tnthk), (viol+, tnth+), (violate, tnth+), etc. using the 4-letter prefix token type.

4 Online Large-Margin Training

Algorithm 1 is our generic online training algorithm. The algorithm is slightly different from other online training algorithms (Tillmann and Zhang, 2006; Liang et al., 2006) in that we keep and update oracle translations, which is a set of good translations reachable by a decoder according to a metric, i.e. BLEU (Papineni et al., 2002). In line 3, a *k*-best list is generated by $\text{best}_k(\cdot)$ using the current weight vector $\mathbf{w}^i$ for the training instance of $(f^t, \mathbf{e}^t)$. Each training instance has multiple (or, possibly one) reference translations $\mathbf{e}^t$ for the source sentence f^t. Using the *k*-best list, *m*-best oracle translations $\mathcal{O}^t$ is updated by $\text{oracle}_m(\cdot)$ for every iteration (line 4). Usually, a decoder cannot generate translations that exactly match the reference translations due to its beam search pruning and OOV. Thus, we cannot always assign scores for each reference translation. Therefore, possible oracle translations are maintained according to an objective function, i.e. BLEU. Tillmann and Zhang (2006) avoided the problem by precomputing the oracle translations in advance. Liang et al. (2006) presented a similar updating strategy in which parameters were updated toward an oracle translation found in C^t, but ignored potentially better translations discovered in the past iterations.

New $\mathbf{w}^{i+1}$ is computed using the k-best list C^t with respect to the oracle translations O^t (line 5). After N iterations, the algorithm returns an averaged weight vector to avoid overfitting (line 9). The key to this online training algorithm is the selection of the updating scheme in line 5.

4.1 Margin Infused Relaxed Algorithm

The Margin Infused Relaxed Algorithm (MIRA) (Crammer et al., 2006) is an online version of the large-margin training algorithm for structured classification (Taskar et al., 2004) that has been successfully used for dependency parsing (McDonald et al., 2005) and joint-labeling/chunking (Shimizu and Haas, 2006). The basic idea is to keep the norm of the updates to the weight vector as small as possible, considering a margin at least as large as the loss of the incorrect classification.

Line 5 of the weight vector update procedure in Algorithm 1 is replaced by the solution of:

$$\begin{aligned}
&\hat{\mathbf{w}}^{i+1} = \operatorname*{argmin}_{\mathbf{w}^{i+1}} ||\mathbf{w}^{i+1} - \mathbf{w}^i|| + C \sum_{\hat{e},e'} \xi(\hat{e}, e')\\
&\text{subject to}\\
&s^{i+1}(f^t, \hat{e}) - s^{i+1}(f^t, e') + \xi(\hat{e}, e') \geq L(\hat{e}, e'; \mathbf{e}^t)\\
&\xi(\hat{e}, e') \geq 0\\
&\forall \hat{e} \in O^t, \forall e' \in C^t \qquad (3)
\end{aligned}$$

where $s^i(f^t, e) = \left\{\mathbf{w}^i\right\}^T \cdot \mathbf{h}(f^t, e)$. $\xi(\cdot)$ is a nonnegative slack variable and $C \geq 0$ is a constant to control the influence to the objective function. A larger C implies larger updates to the weight vector. $L(\cdot)$ is a loss function, for instance difference of BLEU, that measures the difference between $\hat{e}$ and e' according to the reference translations $\mathbf{e}^t$. In this update, a margin is created for each correct and incorrect translation at least as large as the loss of the incorrect translation. A larger error means a larger distance between the scores of the correct and incorrect translations. Following McDonald et al. (2005), only k-best translations are used to form the margins in order to reduce the number of constraints in Eq. 3. In the translation task, multiple translations are acceptable. Thus, margins for m-oracle translation are created, which amount to $m \times k$ large-margin constraints. In this online training, only active features constrained by Eq. 3 are kept and updated, unlike offline training in which all possible features have to be extracted and selected in advance.

The Lagrange dual form of Eq. 3 is:

$$\begin{aligned}
\max_{\alpha(\cdot)\geq 0} \quad & -\frac{1}{2} || \sum_{\hat{e},e'} \alpha(\hat{e}, e') \left(\mathbf{h}(f^t, \hat{e}) - \mathbf{h}(f^t, e')\right) ||^2\\
& + \sum_{\hat{e},e'} \alpha(\hat{e}, e') L(\hat{e}, e'; \mathbf{e}^t)\\
& - \sum_{\hat{e},e'} \alpha(\hat{e}, e') \left(s^i(f^t, \hat{e}) - s^i(f^t, e')\right)\\
\text{subject to} \quad & \sum_{\hat{e},e'} \alpha(\hat{e}, e') \leq C \qquad (4)
\end{aligned}$$

with the weight vector update:

$$\mathbf{w}^{i+1} = \mathbf{w}^i + \sum_{\hat{e},e'} \alpha(\hat{e}, e') \left(\mathbf{h}(f^t, \hat{e}) - \mathbf{h}(f^t, e')\right) \qquad (5)$$

Equation 4 is solved using a QP-solver, such as a coordinate ascent algorithm, by heuristically selecting $(\hat{e}, e')$ and by updating $\alpha(\cdot)$ iteratively:

$$\begin{aligned}
\alpha(\hat{e}, e') &= \max\left(0, \alpha(\hat{e}, e') + \delta(\hat{e}, e')\right) \qquad (6)\\
\delta(\hat{e}, e') &= \frac{L(\hat{e}, e'; \mathbf{e}^t) - \left(s^i(f^t, \hat{e}) - s^i(f^t, e')\right)}{||\mathbf{h}(f^t, \hat{e}) - \mathbf{h}(f^t, e')||^2}
\end{aligned}$$

C is used to clip the amount of updates.

A single oracle with 1-best translation is analytically solved without a QP-solver and is represented as the following perceptron-like update (Shimizu and Haas, 2006):

$$\alpha = \max\left(0, \min\left(C, \frac{L(\hat{e}, e'; \mathbf{e}^t) - \left(s^i(f^t, \hat{e}) - s^i(f^t, e')\right)}{||\mathbf{h}(f^t, \hat{e}) - \mathbf{h}(f^t, e')||^2}\right)\right)$$

Intuitively, the update amount is controlled by the margin and the loss between the correct and incorrect translations and by the closeness of two translations in terms of feature vectors. Indeed, Liang et al. (2006) employed an averaged perceptron algorithm in which α value was always set to one. Tillmann and Zhang (2006) used a different update style based on a convex loss function:

$$\alpha = \eta L(\hat{e}, e'; \mathbf{e}^t) \cdot \max\left(0, 1 - \left(s^i(f^t, \hat{e}) - s^i(f^t, e')\right)\right)$$

Table 1: Experimental results obtained by varying normalized tokens used with surface form.

	# features	2003 (dev)		2004		2005	
		NIST	BLEU [%]	NIST	BLEU [%]	NIST	BLEU [%]
surface form	492K	11.32	54.11	10.57	49.01	10.77	48.05
w/ prefix/suffix	4,204K	12.38	63.87	10.42	48.74	10.58	47.18
w/ word class	2,689K	10.87	49.59	10.63	49.55	10.89	48.79
w/ digits	576K	11.01	50.72	10.66	49.67	10.84	48.39
all token types	13,759K	11.24	52.85	10.66	49.81	10.85	48.41

where $\eta > 0$ is a learning rate for controlling the convergence.

4.2 Approximated BLEU

We used the BLEU score (Papineni et al., 2002) as the loss function computed by:

$$\text{BLEU}(E; \mathbf{E}) = \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log p_n(E, \mathbf{E})\right) \cdot \text{BP}(E, \mathbf{E}) \quad (7)$$

where $p_n(\cdot)$ is the n-gram precision of hypothesized translations $E = \{e^t\}_{t=1}^{T}$ given reference translations $\mathbf{E} = \{\mathbf{e}^t\}_{t=1}^{T}$ and $\text{BP}(\cdot) \leq 1$ is a brevity penalty. BLEU is computed for a set of sentences, not for a single sentence. Our algorithm requires frequent updates on the weight vector, which implies higher cost in computing the document-wise BLEU. Tillmann and Zhang (2006) and Liang et al. (2006) solved the problem by introducing a sentence-wise BLEU. However, the use of the sentence-wise scoring does not translate directly into the document-wise score because of the n-gram precision statistics and the brevity penalty statistics aggregated for a sentence set. Thus, we use an approximated BLEU score that basically computes BLEU for a sentence set, but accumulates the difference for a particular sentence (Watanabe et al., 2006a).

The approximated BLEU is computed as follows: Given oracle translations O for $\mathcal{T}$, we maintain the best oracle translations $O_1^T = \left\{\hat{e}^1, ..., \hat{e}^T\right\}$. The approximated BLEU for a hypothesized translation e' for the training instance $(f^t, \mathbf{e}^t)$ is computed over O_1^T except for $\hat{e}^t$, which is replaced by e':

$$\text{BLEU}(\{\hat{e}^1, ..., \hat{e}^{t-1}, e', \hat{e}^{t+1}, ..., \hat{e}^T\}; \mathbf{E})$$

The loss computed by the approximated BLEU measures the document-wise loss of substituting the correct translation $\hat{e}^t$ into an incorrect translation e'. The score can be regarded as a normalization which scales a sentence-wise score into a document-wise score.

5 Experiments

We employed our online large-margin training procedure for an Arabic-to-English translation task. The training data were extracted from the Arabic/English news/UN bilingual corpora supplied by LDC. The data amount to nearly 3.8M sentences. The Arabic part of the bilingual data is tokenized by isolating Arabic scripts and punctuation marks. The development set comes from the MT2003 Arabic-English NIST evaluation test set consisting of 663 sentences in the news domain with four reference translations. The performance is evaluated by the news domain MT2004/MT2005 test set consisting of 707 and 1,056 sentences, respectively.

The hierarchical phrase translation pairs are extracted in a standard way (Chiang, 2005): First, the bilingual data are word alignment annotated by running GIZA++ (Och and Ney, 2003) in two directions. Second, the word alignment is refined by a grow-diag-final heuristic (Koehn et al., 2003). Third, phrase translation pairs are extracted together with hierarchical phrases by considering holes. In the last step, the hierarchical phrases are constrained so that they follow the target normalized form constraint. A 5-gram language model is trained on the English side of the bilingual data combined with the English Gigaword from LDC.

First, the use of normalized token types in Section 3.3 is evaluated in Table 1. In this setting, all the structural features in Section 3.2 are used, but differentiated by the normalized tokens combined with surface forms. Our online large-margin training algorithm performed 50 iterations constrained

Table 2: Experimental results obtained by incrementally adding structural features.

	# features	2003 (dev)		2004		2005	
		NIST	BLEU [%]	NIST	BLEU [%]	NIST	BLEU [%]
word pairs	11,042K	11.05	51.63	10.43	48.69	10.73	47.72
+ target bigram	11,230K	11.19	53.49	10.40	48.60	10.66	47.47
+ insertion	13,489K	11.21	52.20	10.77	50.33	10.93	48.08
+ hierarchical	13,759K	11.24	52.85	10.66	49.81	10.85	48.41

Table 3: Experimental results for varying *k*-best and *m*-oracle translations.

		# features	2003 (dev)		2004		2005	
			NIST	BLEU [%]	NIST	BLEU [%]	NIST	BLEU [%]
baseline			10.64	46.47	10.83	49.33	10.90	47.03
1-oracle	1-best	8,735K	11.25	52.63	10.82	50.77	10.93	48.11
1-oracle	10-best	10,480K	11.24	53.45	10.55	49.10	10.82	48.49
10-oracle	1-best	8,416K	10.70	47.63	10.83	48.88	10.76	46.00
10-oracle	10-best	13,759K	11.24	52.85	10.66	49.81	10.85	48.41
sentence-BLEU		14,587K	11.10	51.17	10.82	49.97	10.86	47.04

by 10-oracle and 10-best list. When decoding, a 1000-best list is generated to achieve better oracle translations. The training took nearly 1 day using 8 cores of Opteron. The translation quality is evaluated by case-sensitive NIST (Doddington, 2002) and BLEU (Papineni et al., 2002)[2]. The table also shows the number of active features in which non-zero values were assigned as weights. The addition of prefix/suffix tokens greatly increased the number of active features. The setting severely overfit to the development data, and therefore resulted in worse results in open tests. The word class[3] with surface form avoided the overfitting problem. The digit sequence normalization provides a similar generalization capability despite of the moderate increase in the active feature size. By including all token types, we achieved better NIST/BLEU scores for the 2004 and 2005 test sets. This set of experiments indicates that a token normalization is useful especially trained on a small data.

Second, we used all the normalized token types, but incrementally added structural features in Table 2. Target bigram features account for only the fluency of the target side without considering the source/target correspondence. Therefore, the inclusion of target bigram features clearly overfit to the development data. The problem is resolved by adding insertion features which can take into account an agreement with the source side that is not directly captured by word pair features. Hierarchical features are somewhat effective in the 2005 test set by considering the dependency structure of the source side.

Finally, we compared our online training algorithm with sparse features with a baseline system in Table 3. The baseline hierarchical phrase-based system is trained using standard max-BLEU training (MERT) without sparse features (Och, 2003). Table 3 shows the results obtained by varying the *m*-oracle and *k*-best size ($k, m = 1, 10$) using all structural features and all token types. We also experimented sentence-wise BLEU as an objective function constrained by 10-oracle and 10-best list. Even the 1-oracle 1-best configuration achieved significant improvements over the baseline system. The use of a larger *k*-best list further optimizes to the development set, but at the cost of degraded translation quality in the 2004 test set. The larger *m*-oracle size seems to be harmful if coupled with the 1-best list. As indicated by the reduced active feature size, 1-best translation seems to be updated toward worse translations in 10-oracles that are "close" in terms of features. We achieved significant improvements

[2] We used the tool available at http://www.nist.gov/speech/tests/mt/

[3] We induced 50 classes each for English and Arabic.

Table 4: Two-fold cross validation experiments.

	closed test		open test	
	NIST	BLEU [%]	NIST	BLEU [%]
baseline	10.71	44.79	10.68	44.44
online	11.58	53.42	10.90	47.64

when the *k*-best list size was also increased. The use of sentence-wise BLEU as an objective provides almost no improvement in the 2005 test set, but is comparable for the 2004 test set.

As observed in three experiments, the 2004/2005 test sets behaved differently, probably because of the domain mismatch. Thus, we conducted a two-fold cross validation using the 2003/2004/2005 test sets to observe the effect of optimization as shown in Table 4[4]. The MERT baseline system performed similarly both in closed and open tests. Our online large-margin training with 10-oracle and 10-best constraints and the approximated BLEU loss function significantly outperformed the baseline system in the open test. The development data is almost doubled in this setting. The MERT approach seems to be confused with the slightly larger data and with the mixed domains from different epochs.

6 Discussion

In this work, the translation model consisting of millions of features are successfully integrated. In order to avoid poor overfitting, features are limited to word-based features, but are designed to reflect the structures inside hierarchical phrases. One of the benefit of MIRA is its flexibility. We may include as many constraints as possible, like *m*-oracle constraints in our experiments. Although we described experiments on the hierarchical phrase-based translation, the online training algorithm is applicable to any translation systems, such as phrase-based translations and syntax-based translations.

Online discriminative training has already been studied by Tillmann and Zhang (2006) and Liang et al. (2006). In their approach, training was performed on a large corpus using the sparse features of phrase translation pairs, target *n*-grams and/or bag-of-word pairs inside phrases. In Tillmann and Zhang (2006), *k*-best list generation is approximated by a step-by-step one-best merging method that separates the decoding and training steps. The weight vector update scheme is very similar to MIRA but based on a convex loss function. Our method directly employs the *k*-best list generated by the fast decoding method (Watanabe et al., 2006b) at every iteration. One of the benefits is that we avoid the rather expensive cost of merging the *k*-best list especially when handling millions of features.

Liang et al. (2006) employed an averaged perceptron algorithm. They decoded each training instance and performed a perceptron update to the weight vector. An incorrect translation was updated toward an oracle translation found in a *k*-best list, but discarded potentially better translations in the past iterations.

An experiment has been undertaken using a small development set together with sparse features for the reranking of a *k*-best translation (Watanabe et al., 2006a). They relied on a variant of a voted perceptron, and achieved significant improvements. However, their work was limited to reranking, thus the improvement was relative to the performance of the baseline system, whether or not there was a good translation in a list. In our work, the sparse features are directly integrated into the DP-based search.

The design of the sparse features was inspired by Zens and Ney (2006). They exploited the word alignment structure inside the phrase translation pairs for discriminatively training a reordering model in their phrase-based translation. The reordering model simply classifies whether to perform monotone decoding or not. The trained model is treated as a single feature function integrated in Eq. 1. Our approach differs in that each sparse feature is individually integrated in Eq. 1.

7 Conclusion

We exploited a large number of binary features for statistical machine translation. The model was trained on a small development set. The optimization was carried out by MIRA, which is an online version of the large-margin training algorithm. Millions of sparse features are intuitively considered prone to overfitting, especially when trained on a small development set. However, our algorithm with

[4] We split data by document, not by sentence.

millions of features achieved very significant improvements over a conventional method with a small number of features. This result indicates that we can easily experiment many alternative features even with a small data set, but we believe that our approach can scale well to a larger data set for further improved performance. Future work involves scaling up to larger data and more features.

Acknowledgements

We would like to thank reviewers and our colleagues for useful comment and discussion.

References

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. 2006. Sequence classification for machine translation. In *Proc. of Interspeech 2006*, pages 1157–1160, Pittsburgh.

Oliver Bender, Richard Zens, Evgeny Matusov, and Hermann Ney. 2004. Alignment templates: the RWTH SMT system". In *Proc. of IWSLT 2004*, pages 79–84, Kyoto, Japan.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL 2005*, pages 263–270, Ann Arbor, Michigan, June.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, March.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *In Proc. ARPA Workshop on Human Language Technology*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of COLING/ACL 2006*, pages 961–968, Sydney, Australia, July.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL 2003*, pages 48–54, Edmonton, Canada.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of COLING/ACL 2006*, pages 761–768, Sydney, Australia, July.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL 2005*, pages 91–98, Ann Arbor, Michigan, June.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL 2002*, pages 311–318, Philadelphia, Pennsylvania.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. of ACL 2004*, pages 47–54, Barcelona, Spain, July.

Nobuyuki Shimizu and Andrew Haas. 2006. Exact decoding for jointly labeling and chunking sequences. In *Proc. of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 763–770, Sydney, Australia, July.

Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proc. of EMNLP 2004*, pages 1–8, Barcelona, Spain, July.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proc. of COLING/ACL 2006*, pages 721–728, Sydney, Australia, July.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2006a. NTT Statistical Machine Translation for IWSLT 2006. In *Proc. of IWSLT 2006*, pages 95–102, Kyoto, Japan.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006b. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of COLING/ACL 2006*, pages 777–784, Sydney, Australia, July.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proc. of COLING 98*, pages 1408–1415, Montreal, Quebec, Canada.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proc. of WSMT 2006*, pages 55–63, New York City, June.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of WSMT 2006*, pages 138–141, New York City, June.

Scalable Term Selection for Text Categorization

Jingyang Li
National Lab of Intelligent Tech. & Sys.
Department of Computer Sci. & Tech.
Tsinghua University, Beijing, China
lijingyang@gmail.com

Maosong Sun
National Lab of Intelligent Tech. & Sys.
Department of Computer Sci. & Tech.
Tsinghua University, Beijing, China
sms@tsinghua.edu.cn

Abstract

In text categorization, term selection is an important step for the sake of both categorization accuracy and computational efficiency. Different dimensionalities are expected under different practical resource restrictions of time or space. Traditionally in text categorization, the same scoring or ranking criterion is adopted for all target dimensionalities, which considers both the discriminability and the coverage of a term, such as χ^2 or IG. In this paper, the poor accuracy at a low dimensionality is imputed to the small average vector length of the documents. Scalable term selection is proposed to optimize the term set at a given dimensionality according to an expected average vector length. Discriminability and coverage are separately measured; by adjusting the ratio of their weights in a combined criterion, the expected average vector length can be reached, which means a good compromise between the specificity and the exhaustivity of the term subset. Experiments show that the accuracy is considerably improved at lower dimensionalities, and larger term subsets have the possibility to lower the average vector length for a lower computational cost. The interesting observations might inspire further investigations.

1 Introduction

Text categorization is a classical text information processing task which has been studied adequately (Sebastiani, 2002). A typical text categorization process usually involves these phases: document indexing, dimensionality reduction, classifier learning, classification and evaluation. The vector space model is frequently used for text representation (document indexing); dimensions of the learning space are called *terms*, or *features* in a general machine learning context. Term selection is often necessary because:

- Many *irrelevant* terms have detrimental effect on categorization accuracy due to *overfitting* (Sebastiani, 2002).
- Some text categorization tasks have many relevant but *redundant* features, which also hurt the categorization accuracy (Gabrilovich and Markovitch, 2004).
- Considerations on computational cost:
  (i) Many sophisticated learning machines are very slow at high dimensionalities, such as LLSF (Yang and Chute, 1994) and SVMs. (ii) In Asian languages, the term set is often very large and redundant, which causes the learning and the predicting to be really slow. (iii) In some practical cases the computational resources (time or space) are restricted, such as hand-held devices, real-time applications and frequently retrained systems. (iv) Some deeper analysis or feature reconstruction techniques rely on matrix factorization (e.g. LSA based on SVD), which might be computationally intractable while the dimensionality is large.

Sometimes an aggressive term selection might be needed particularly for (iii) and (iv). But it is notable that the dimensionality is not always directly

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 774–782, Prague, June 2007. ©2007 Association for Computational Linguistics

connected to the computational cost; this issue will be touched on in Section 6. Although we have many general feature selection techniques, the domain specified ones are preferred (Guyon and Elisseeff, 2003). Another reason for ad hoc *term* selection techniques is that many other pattern classification tasks has no *sparseness problem* (in this study the sparseness means a sample vector has few nonzero elements, but not the high-dimensional learning space has few training samples). As a basic motivation of this study, we hypothesize that the low accuracy at low dimensionalities is mainly due to the sparseness problem.

Many term selection techniques were presented and some of them have been experimentally tested to be high-performing, such as Information Gain, χ^2 (Yang and Pedersen, 1997; Rogati and Yang, 2002) and Bi-Normal Separation (Forman, 2003). Everyone of them adopt a criterion scoring and ranking the terms; for a target dimensionality d, the term selection is simply done by picking out the top-d terms from the ranked term set. These high performing criteria have a common characteristic — both discriminability and coverage are implicitly considered.

- *discriminability*: how unbalanced is the distribution of the term among the categories.
- *coverage*: how many documents does the term occur in.

(Borrowing the terminologies from document indexing, we can say the *specificity* of a term set corresponds to the discriminability of each term, and the *exhaustivity* of a term set corresponds to the coverage of each term.) The main difference among these criteria is to what extent the discriminability is emphasized or the coverage is emphasized. For instance, empirically *IG* prefers high frequency terms more than χ^2 does, which means *IG* emphasizes the coverage more than χ^2 does.

The problem is, these criteria are nonparametric and do the same ranking for any target dimensionality. Small term sets meet the specificity–exhaustivity dilemma. If really the sparseness is the main reason of the low performance of a small term set, the specificity should be moderately sacrificed to improve the exhaustivity for a small term set; that is to say, the term selection criterion should consider coverage more than discriminability. Contrariwise, coverage could be less considered for a large term set, because we need worry little about the sparseness problem and the computational cost might decrease.

The remainder of this paper is organized as follows: Section 2 describes the document collections used in this study, as well as other experiment settings; Section 3 investigates the relation between sparseness (measured by *average vector length*) and categorization accuracy; Section 4 explains the basic idea of scalable term selection and proposed a potential approach; Section 5 carries out experiments to evaluate the approach, during which some empirical rules are observed to complete the approach; Section 6 makes some further observations and discussions based on Section 5; Section 7 gives a concluding remark.

2 Experiment Settings

2.1 Document Collections

Two document collections are used in this study.

CE (Chinese Encyclopedia): This is from the electronic version of the Chinese Encyclopedia. We choose a Chinese corpus as the primary document collection because Chinese text (as well as other Asian languages) has a very large term set and a satisfying subset is usually not smaller than 50000 (Li et al., 2006); on the contrary, a dimensionality lower than 10000 suffices a general English text categorization (Yang and Pedersen, 1997; Rogati and Yang, 2002). For computational cost reasons mentioned in Section 1, Chinese text categorization would benefit more from an high-performing aggressive term selection. This collection contains 55 categories and 71674 documents (9:1 split to training set and test set). Each documents belongs to only one category. Each category contains 399–3374 documents. This collection was also used by Li et al. (2006).

20NG (20 Newsgroups[1]): This classical English document collection is chosen as a secondary in this study to testify the generality of the proposed approach. Some figures about this collection are not shown in this paper as the figures about CE, viz. Figure 1–4 because they are similar to CE's.

[1] http://people.csail.mit.edu/jrennie/20Newsgroups

2.2 Other Settings

For CE collection, character bigrams are chosen to be the indexing unit for its high performance (Li et al., 2006); but the bigram term set suffers from its high dimensionality. This is exactly the case we tend to tackle. For 20NG collection, the indexing units are stemmed[2] words. Both term set are *df*-cut by the most conservative threshold ($df \geq 2$). The sizes of the two candidate term sets are $|\mathcal{T}_{\text{CE}}| = 1067717$ and $|\mathcal{T}_{\text{20NG}}| = 30220$.

Term weighting is done by $\textit{tfidf}(t_i, d_j) = \log(\textit{tf}(t_i, d_j) + 1) \cdot \log\left(\frac{\textit{df}(t_i)+1}{N_d}\right)$[3], in which t_i denotes a term, d_j denotes a document, N_d denotes the total document number.

The classifiers used in this study are support vector machines (Joachims, 1998; Gabrilovich and Markovitch, 2004; Chang and Lin, 2001). The kernel type is set to linear, which is fast and enough for text categorization. Also, Brank et al. (2002) pointed out that the complexity and sophistication of the criterion itself is more important to the success of the term selection method than its compatibility in design with the classifier.

Performance is evaluated by microaveraged F_1-measure. For single-label tasks, microaveraged *precision*, *recall* and F_1 have the same value.

χ^2 is used as the term selection baseline for its popularity and high performance. (*IG* was also reported to be good. In our previous experiments, χ^2 is generally superior to *IG*.) In this study, features are always selected *globally*, which means the maximum are computed for category-specific values (Sebastiani, 2002).

3 Average Vector Length (AVL)

In this study, *vector length* (how many different terms does the document hold after term selection) is used as a straightforward sparseness measure for a document (Brank et al., 2002). Generally, document sizes have a *lognormal distribution* (Mitzenmacher, 2003). In our experiment, vector lengths are also found to be nearly lognormal distributed, as shown in Figure 1. If the correctly classified documents and the wrongly classified documents are separately investigated, they both yield a nearly lognormal distribution.

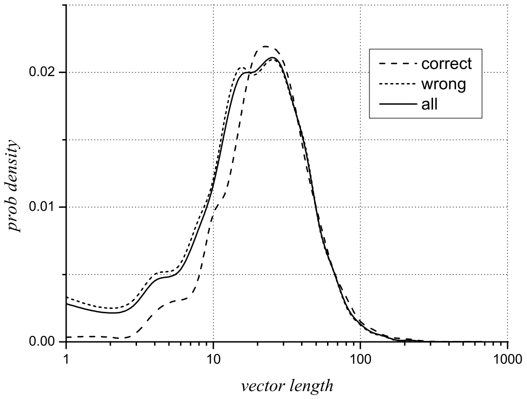


Figure 1: Vector Length Distributions (smoothed), on CE Document Collection

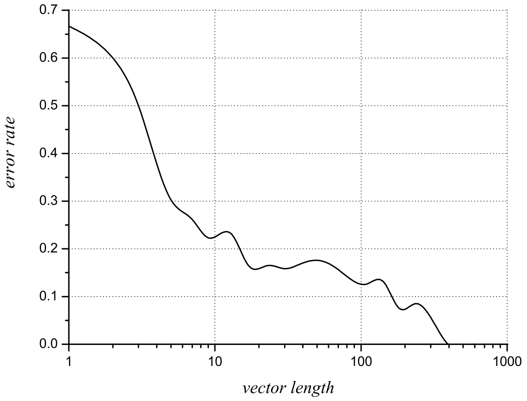


Figure 2: Error Rate vs. Vector Length (smoothed), on CE Collection, 5000 Dimensions by χ^2

Also in Figure 1, wrongly classified documents shows a relatively large proportion at low dimensionalities. Figure 2 demonstrates this with more clarity. Thus the hypothesis formed in Section 1 is confirmed: there is a strong correlation between the sparseness degree and the categorization error rate.

Therefore, it is quite straightforward a thought to measure the “sparseness of a term subset” (or more precisely, the exhaustivity) by the corresponding *average vector length* (*AVL*) of all documents.[4] In the

[2] Stemming by Porter's Stemmer (`http://www.tartarus.org/ martin/PorterStemmer/`).

[3] In our experiments this form of *tfidf* always outperforms the basic $\textit{tfidf}(t_i, d_j) = \textit{tf}(t_i, d_j) \cdot \log\left(\frac{\textit{df}(t_i)+1}{N_d}\right)$ form.

[4] Due to the lognormal distribution of vector length, it seems more plausible to average the logarithmic vector length. However, for a fixed number of documents , $\log \frac{\sum |d_j|}{|\mathcal{D}|}$ should hold a nearly fixed ratio to $\frac{\sum \log |d_j|}{|\mathcal{D}|}$, in which $|\mathcal{D}|$ denotes the document number and $|d_j|$ denotes the document vector length.

remainder of this paper, (log) *AVL* is an important metric used to assess and control the sparseness of a term subset.

4 Scalable Term Selection (STS)

Since the performance droping down at low dimensionalities is attributable to low *AVL*s in the previous section, a scalable term selection criterion should *automatically* accommodate its favor of high coverage to different target dimensionalities.

4.1 Measuring Discriminability and Coverage

The first step is to separately measure the discriminability and the coverage of a term. A basic guideline is that these two metrics should not be highly (positive) correlated; intuitively, they should have a slight negative correlation. The correlation of the two metrics can be visually estimated by the joint distribution figure. A bunch of term selection metrics were explored by Forman (2003). *df* (*document frequency*) is a straightforward choice to measure coverage. Since *df* follows the Zipf's law (inverse power law), $\log(df)$ is adopted. High-performing term selection criterion themselves might not be good candidates for the discriminability metric because they take coverage into account. For example, Figure 3 shows that χ^2 is not satisfying. (For readability, the grayness is proportional to the log probability density in Figure 3, Figure 4 and Figure 12.) Relatively, *probability ratio* (Forman, 2003) is a more straight metric of discriminability.

$$PR(t_i, c) = \frac{P(t_i|c_+)}{P(t_i|c_-)} = \frac{df(t_i, c_+)/df(c_+)}{df(t_i, c_-)/df(c_-)}$$

It is a symmetric ratio, so $\log(PR)$ is likely to be more appropriate. For multi-class categorization, a global value can be assessed by $PR_{\max}(t_i) = \max_c PR(t_i, c)$, like $\chi^2_{\max}$ for χ^2 (Yang and Pedersen, 1997; Rogati and Yang, 2002; Sebastiani, 2002); for brief, *PR* denotes $PR_{\max}$ hereafter. The joint distribution of $\log(PR)$ and $\log(df)$ is shown in Figure 12. We can see that the distribution is quite even and they have a slight negative correlation.

4.2 Combined Criterion

Now we have the two metrics: $\log(PR)$ for discriminability and $\log(df)$ for coverage, and a parametric term selection criterion comes forth:

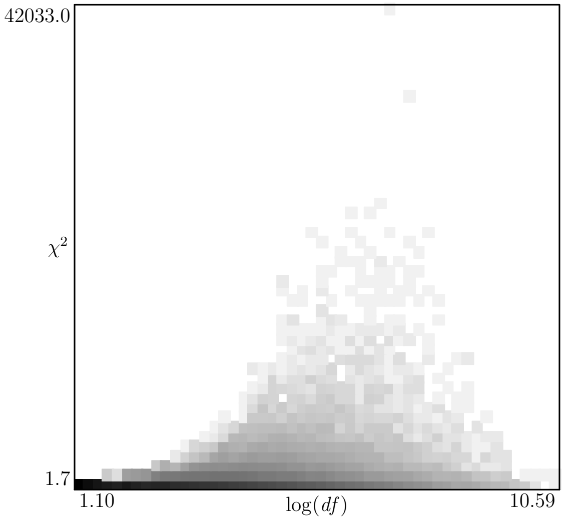


Figure 3: $\left(\log(df), \chi^2\right)$ Distribution, on CE

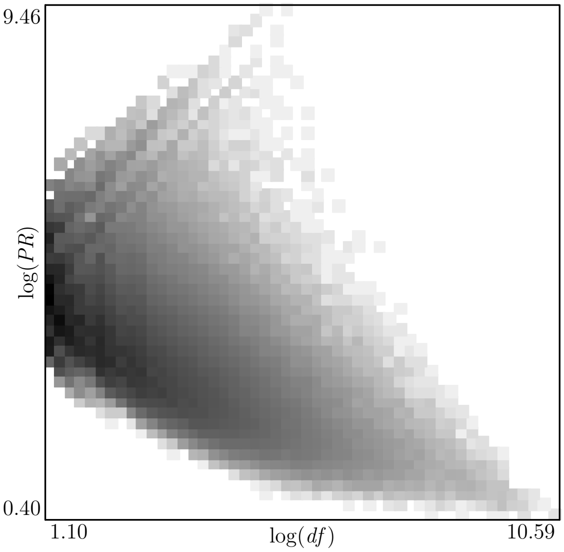


Figure 4: $(\log(df), \log(PR))$ Distribution, on CE

$$\zeta(t_i; \lambda) = \left(\frac{\lambda}{\log(PR(t_i))} + \frac{1-\lambda}{\log(df(t_i))}\right)^{-1}$$

A *weighted harmonic averaging* is adopted here because either metric's being *too small* is a severe detriment. $\lambda \in [0, 1]$ is the weight for $\log(PR)$, which denotes how much the discriminability is emphasized. When the dimensionality is fixed, a smaller λ leads to a larger *AVL* and a larger λ leads to a smaller *AVL*. The optimal λ should be a function of the expected dimensionality (k):

$$\lambda^*(k) = \arg\max_{\lambda} F_1(\mathcal{S}_k(\lambda))$$

in which the term subset $\mathcal{S}_k(\lambda) \in \mathcal{T}$ is selected by $\zeta(\circ;\lambda)$, $|\mathcal{S}_k| = k$, and F_1 is the default evaluation criterion. Naturally, this optimal λ leads to a corresponding optimal *AVL*:

$$\mathit{AVL}^*(k) \longleftrightarrow \lambda^*(k)$$

For a concrete implementation, we should have an (empirical) function to estimate λ^* or *AVL**:

$$\mathit{AVL}^\circ(k) \doteq \mathit{AVL}^*(k)$$

In the next section, the values of *AVL** (as well as λ^*) for some k-s are figured out by experimental search; then an empirical formula, *AVL*$^\circ(k)$, comes forth. It is interesting and inspiring that by adding the "corpus *AVL*" as a parameter this formula is universal for different document collections, which makes the whole idea valuable.

5 Experiments and Implementation

5.1 Experiments

The expected dimensionalities (k) chosen for experimentation are
CE: 500, 1000, 2000, 4000, . . . , 32000, 64000;
20NG: 500, 1000, 2000, . . . , 16000, 30220.[5]

For a given document collection and a given target dimensionality, there is a corresponding *AVL* for a λ, and vice versa (for the possible value range of *AVL*). According to the observations in Section 5.2, *AVL* other than λ is the direct concern because it is more intrinsic, but λ is the one that can be tuned directly. So, in the experiments, we vary *AVL* by tuning λ to produce it, which means to calculate $\lambda(\mathit{AVL})$.

AVL(λ) is a monotone function and fast to calculate. For a given *AVL*, the corresponding λ can be quickly found by a Newton iteration in [0,1]. In fact, *AVL*(λ) is not a continuous function, so λ is only tuned to get an acceptable match, e.g. within ± 0.1.

For each k, by the above way of fitting λ, we manually adjust *AVL* (only in integers) until $F_1(\mathcal{S}_k(\lambda(\mathit{AVL})))$ peaks. By this way, Figure 5–11 are manually tuned best-performing results as observations for figuring out the empirical formulas.

Figure 5 shows the F_1 peaks at different dimensionalities. Comparing to χ^2, STS has a considerable potential superiority at low dimensionalities. The corresponding values of *AVL** are shown in Figure 6, along with the *AVL*s of χ^2-selected term subsets. The dotted lines show the trend of *AVL**; at the overall dimensionality, $|\mathcal{T}_{\text{CE}}| = 1067717$, they have the same *AVL* $= 898.5$. We can see that $\log(\mathit{AVL}^*)$ is almost proportional to $\log(k)$ when k is not too large. The corresponding values of λ^* are shown in Figure 7; the relation is nearly linear between λ^* and $\log(k)$.

Now it is necessary to explain why an empirical *AVL*$^\circ(k)$ derived from the straight line in Figure 6 can be used instead of *AVL*$^*(k)$ in practice. One important but not plotted property is that the performance of STS is not very sensitive to a small value change of *AVL*. For instance, at $k = 4000$, *AVL*$^* = 120$ and the F_1 peak is 85.8824%, and for *AVL* $= 110$ and 130 the corresponding F_1 are 85.8683% and 85.6583%; at the same k, the F_1 of χ^2 selection is 82.3950%. This characteristic of STS guarantee that the empirical *AVL*$^\circ(k)$ has a very close performance to *AVL*$^*(k)$; due to the limited space, the performance curve of *AVL*$^\circ(k)$ will not be plotted in Section 5.2.

Same experiments are done on 20NG and the results are shown in Figure 8, Figure 9 and Figure 10. The performance improvements is not as significant as on the CE collection; this will be discussed in Section 6.2. The conspicuous relations between *AVL**, λ^* and k remain the same.

5.2 Algorithm Completion

In Figure 6 and Figure 9, the ratios of $\log(\mathit{AVL}^*(k))$ to $\log(k)$ are not the same on CE and 20NG. Taking into account the *corpus AVL* (the *AVL* produced by the whole term set): $\mathit{AVL}_{\mathcal{T}_{\text{CE}}} = 898.5286$ and $\mathit{AVL}_{\mathcal{T}_{\text{20NG}}} = 82.1605$, we guess $\frac{\log(\mathit{AVL}^*(k))}{\log(\mathit{AVL}_{\mathcal{T}})}$ is capable of keeping the same ratio to $\log(k)$ for both CE and 20NG. This hypothesis is confirmed (not for too high dimensionalities) by Figure 11; Section 6.2

[5] STS is tested to the whole $\mathcal{T}$ on 20NG but not on CE, because (i) $\mathcal{T}_{\text{CE}}$ is too large and time consuming for training and testing, and (ii) χ^2 was previously tested on larger k and the performance (F_1) is not stable while $k > 64000$.

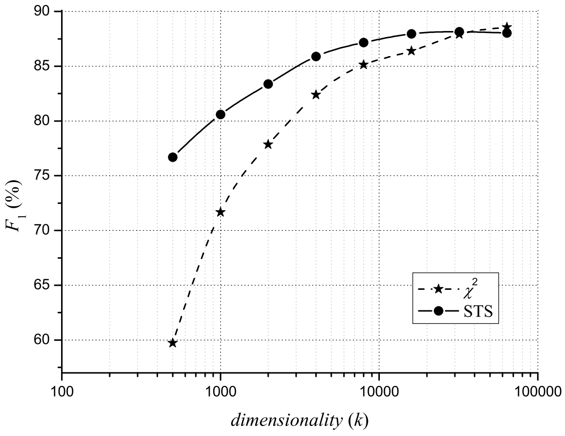


Figure 5: Performance Comparison, on CE

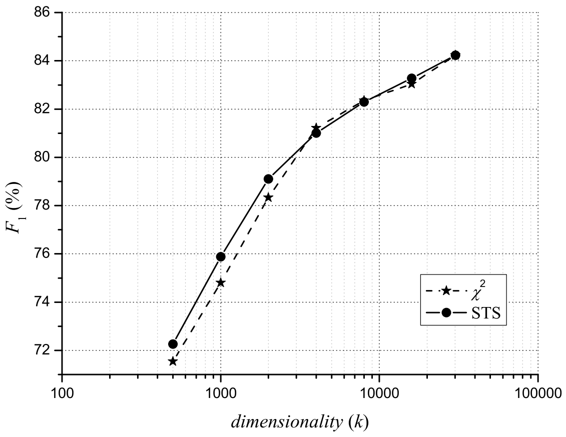


Figure 8: Performance Comparison, on 20NG

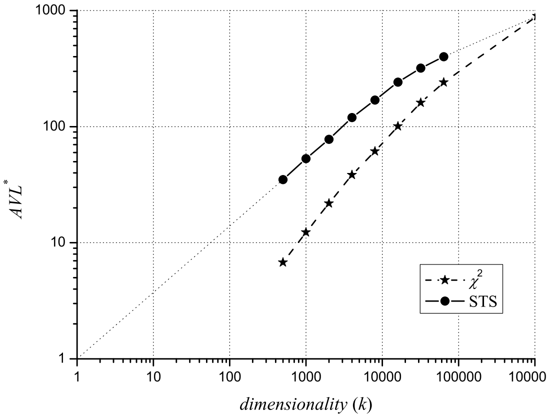


Figure 6: *AVL* Comparison, on CE

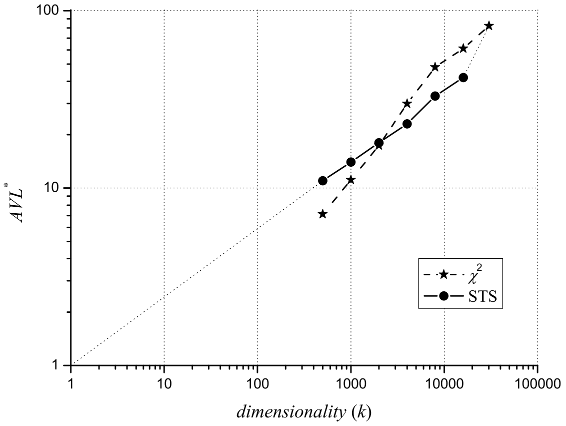


Figure 9: *AVL* Comparison, on 20NG

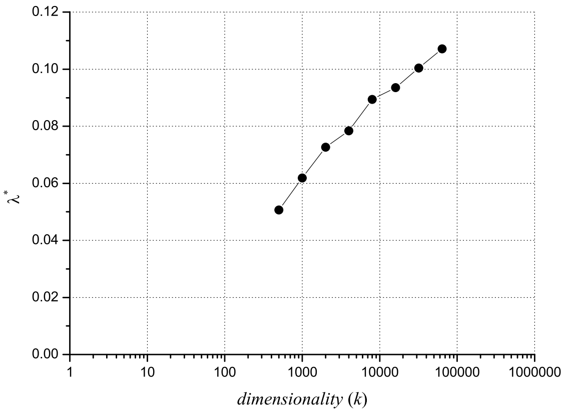


Figure 7: Optimal Weights of $\log(PR)$, on CE

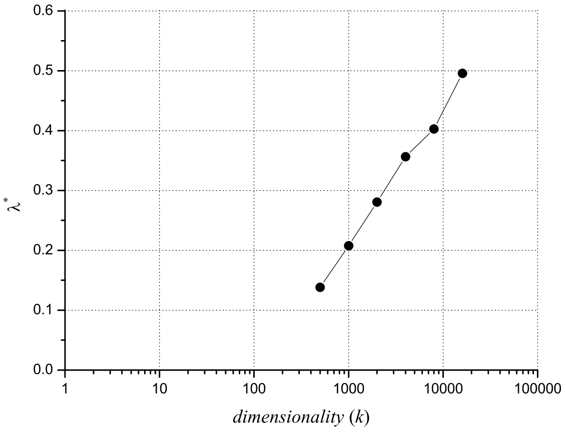


Figure 10: Optimal Weights of $\log(PR)$, on 20NG

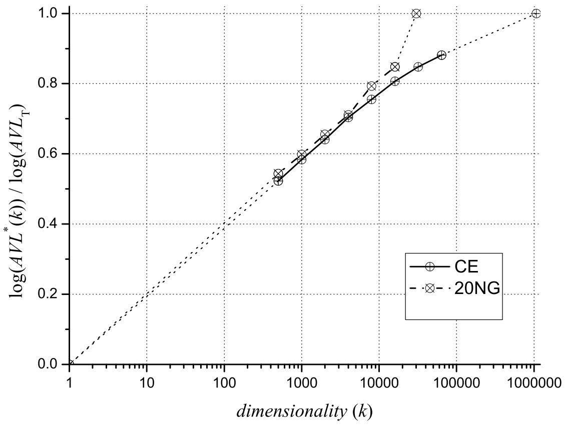


Figure 11: $\frac{\log(\mathit{AVL}^*(k))}{\log(\mathit{AVL}_{\mathcal{T}})}$, on Both CE and 20NG

contains some discussion on this.

From the figure, we get the value of this ratio (the base of $\log$ is set to e):

$$\gamma = \frac{\log(\mathit{AVL}^*(k))/\log(\mathit{AVL}_{\mathcal{T}})}{\log(k)} \cong 0.085$$

which should be a universal constant for all text categorization tasks.

So the empirical estimation of $\mathit{AVL}^*(k)$ is given by

$$\begin{aligned}\mathit{AVL}^{\circ}(k) &= \exp(\gamma \log(\mathit{AVL}_{\mathcal{T}}) \cdot \log(k)) \\ &= \mathit{AVL}_{\mathcal{T}}^{\gamma \log(k)}\end{aligned}$$

and the final STS criterion is

$$\begin{aligned}\zeta(t_i, k) &= \zeta(t_i; \lambda(\mathit{AVL}^{\circ}(k))) \\ &= \zeta(t_i; \lambda(\mathit{AVL}_{\mathcal{T}}^{\gamma \log(k)}))\end{aligned}$$

in which $\lambda(\circ)$ can be calculated as in Section 5.1. The target dimensionality, k, is involved as a parameter, so the approach is named *scalable* term selection. As stated in Section 5.1, $\mathit{AVL}^{\circ}(k)$ has a very close performance to $\mathit{AVL}^*(k)$ and its performance is not plotted here.

6 Further Observation and Discussion

6.1 Comparing the Selected Subsets

An investigation shows that for a quite large range of λ, term rankings by $\zeta(t_i; \lambda)$ and $\chi^2(t_i)$ have a strong correlation (the *Spearman's rank correlation coefficient* is bigger than 0.999). In order to compare the two criteria's preferences for discriminability and coverage, the selected subsets of different dimensionalities are shown in Figure 12 (the corresponding term density distribution was shown in Figure 4) and Figure 13. For different dimension-

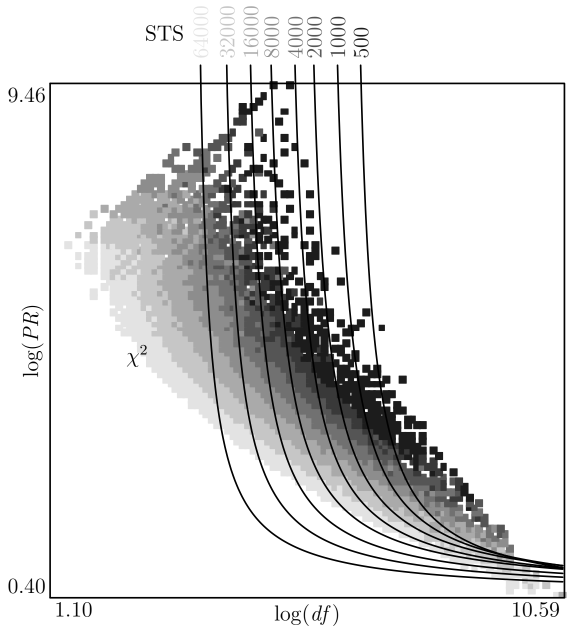


Figure 12: Selection Area Comparison of STS and χ^2 on Various Dimensionalities, on CE

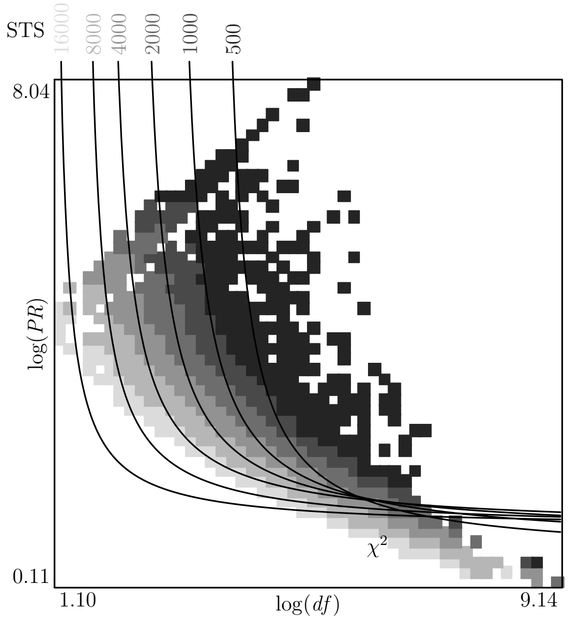


Figure 13: Selection Area Comparison of STS and χ^2 on Various Dimensionalities, on 20NG

alities, the selection areas of STS are represented by boundary lines, and the selection areas of χ^2 are represented by different grayness.

In Figure 12, STS shows its superiority at low dimensionalities by more emphasis on the coverage of terms. In Figure 13, STS shows its superiority at high dimensionalities by more emphasis on the discriminability of terms; lower coverage yields smaller index size and lower computational cost. At any dimensionality, STS yields a relatively fixed bound for either discriminability or coverage, other than a compromise between them like χ^2; this is attributable to the harmonic averaging.

6.2 Adaptability of STS

There are actually two kinds of sparseness in a (vectorized) document collection:

collection sparseness: the high-dimensional learning space contains few training samples;

document sparseness: a document vector has few nonzero dimensions.

In this study, only the document sparseness is investigated. The collection sparseness might be a backroom factor influencing the actual performance on different document collections. This might explain why the explicit characteristics of STS are not the same on CE to 20NG: (comparing with χ^2, see Figure 5, Figure 6, Figure 8 and Figure 9)

CE. The significant F_1 improvements at low dimensionalities sacrifice the short of *AVL*. In some learning process implementations, it is *AVL* other than k that determines the computational cost; in many other cases, k is the determinant. Further more, possible post-processing, like matrix factorization, might benefit from a low k.

20NG. The F_1 improvements at low dimensionalities is not quite significant, but *AVL* remains a lower level. For higher k, there is less difference in F_1, but the smaller *AVL* yield lower computational cost than χ^2.

Nevertheless, STS shows a stable behavior for various dimensionalities and quite different document collections. The existence of the universal constant γ empowers it to be adaptive and practical. As shown in Figure 11, STS draws the *relative* $\log \mathit{AVL}^*(k)$ to the same straight line, $\gamma \log(k)$, for different document collections. This might means that the *relative AVL* is an intrinsic demand for the term subset size k.

7 Conclusion

In this paper, Scalable Term Selection (STS) is proposed and supposed to be more adaptive than traditional high-performing criteria, viz. χ^2, *IG*, *BNS*, etc. The basic idea of STS is to separately measure discriminability and coverage, and adjust the relative importance between them to produce a optimal term subset of a given size. Empirically, the constant relation between target dimensionality and the optimal relative average vector length is found, which turned the idea into implementation.

STS showed considerable adaptivity and stability for various dimensionalities and quite different document collections. The categorization accuracy increasing at low dimensionalities and the computational cost decreasing at high dimensionalities were observed.

Some observations are notable: the loglinear relation between optimal average vector length (*AVL**) and dimensionality (k), the semi-loglinear relation between weight λ and dimensionality, and the universal constant γ. For a future work, STS needs to be conducted on more document collections to check if γ is really universal.

In addition, there could be other implementations of the general STS idea, via other metrics of discriminability and coverage, other weighted combination forms, or other term subset evaluations.

Acknowledgement

The research is supported by the National Natural Science Foundation of China under grant number 60573187, 60621062 and 60520130299.

References

Janez Brank, Marko Grobelnik, Nataša Milic-Fraylingand, and Dunjia Mladenic. 2002. Interaction of feature selection methods and linear classification models. *Workshop on Text Learning held at ICML-2002*.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305.

Evgeniy Gabrilovich and Shaul Markovitch. 2004. Text categorization with many redundant features: using aggressive feature selection to make svms competitive with c4.5. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 41, New York, NY, USA. ACM Press.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Thorsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML '98*, number 1398, pages 137–142. Springer Verlag, Heidelberg, DE.

Jingyang Li, Maosong Sun, and Xian Zhang. 2006. A comparison and semi-quantitative analysis of words and character-bigrams as features in chinese text categorization. In *Proceedings of COLING-ACL '06*, pages 545–552. Association for Computational Linguistics, July.

Michael Mitzenmacher. 2003. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1:226–251.

Monica Rogati and Yiming Yang. 2002. High-performing feature selection for text classification. In *Proceedings of CIKM '02*, pages 659–661. ACM Press.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47.

Yiming Yang and Christopher G. Chute. 1994. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems (TOIS)*, 12(3):252–277.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US. Morgan Kaufmann Publishers, San Francisco, US.

Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem

Jingbo Zhu
University of Southern California
Information Sciences Institute
Northeastern University, P.R.China
Natural Language Processing Lab
Zhujingbo@mail.neu.edu.cn

Eduard Hovy
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
hovy@isi.edu

Abstract

In this paper, we analyze the effect of resampling techniques, including under-sampling and over-sampling used in active learning for word sense disambiguation (WSD). Experimental results show that under-sampling causes negative effects on active learning, but over-sampling is a relatively good choice. To alleviate the within-class imbalance problem of over-sampling, we propose a bootstrap-based over-sampling (BootOS) method that works better than ordinary over-sampling in active learning for WSD. Finally, we investigate when to stop active learning, and adopt two strategies, max-confidence and min-error, as stopping conditions for active learning. According to experimental results, we suggest a prediction solution by considering max-confidence as the upper bound and min-error as the lower bound for stopping conditions.

1 Introduction

Word sense ambiguity is a major obstacle to accurate information extraction, summarization, and machine translation (Ide and Veronis, 1998). In recent years, a variety of techniques for machine learning algorithms have demonstrated remarkable performance for automated word sense disambiguation (WSD) (Chan and Ng, 2006; Dagan et. al., 2006; Xue *et. al.*, 2006; Kohomban and Lee. 2005; Dang and Palmer, 2005), when enough labeled training data is available. However, creating a large sense-tagged corpus is very expensive and time-consuming, because these data have to be annotated by human experts.

Among the techniques to solve the knowledge bottleneck problem, active learning is a promising way (Lewis and Gale, 1994; McCallum and Nigram, 1998). The purpose of active learning is to minimize the amount of human labeling effort by having the system automatically select for human annotation the most informative unannotated case.

In real-world data, the distribution of the senses of a word is often very skewed. Some studies reported that simply selecting the predominant sense provides superior performance, when a highly skewed sense distribution and insufficient context exist (Hoste *et al.*, 2001; McCarthy *et. al.*, 2004). The data set is *imbalanced* when at least one of the senses is heavily underrepresented compared to the other senses. In general, a WSD classifier is designed to optimize overall accuracy without taking into account the class imbalance distribution in a real-world data set. The result is that the classifier induced from imbalanced data tends to over-fit the predominant class and to ignore small classes (Japkowicz and Stephen, 2002). Recently, much work has been done in addressing the class imbalance problem, reporting that resampling methods such as *over-sampling* and *under-sampling* are useful in supervised learning with imbalanced data sets to induce more effective classifiers (Estabrooks *et al.*, 2004; Zhou and Liu, 2006).

In general framework of active learning, the learner (i.e. supervised classifier) is formed by using supervised learning algorithms. To date, however, no-one has studied the effects of over-sampling and under-sampling on active learning

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 783–790, Prague, June 2007. ©2007 Association for Computational Linguistics

methods. In this paper, we study active learning with resampling methods addressing the class imbalance problem for WSD. It is noteworthy that neither of these techniques need modify the architecture or learning algorithm, making them very easy to use and extend to other domains.

Another problem in active learning is knowing when to stop the process. We address this problem in this paper, and discuss how to form the final classifier for use. This is a problem of estimation of classifier effectiveness (Lewis and Gale, 1994). Because it is difficult to know when the classifier reaches maximum effectiveness, previous work used a simple stopping condition when the training set reaches desirable size. However, in fact it is almost impossible to predefine an appropriate size of desirable training data for inducing the most effective classifier. To solve the problem, we consider the problem of estimation of classifier effectiveness as a second task of estimating classifier confidence. This paper adopts two strategies: *max-confidence* and *min-error*, and suggests a prediction solution by considering max-confidence as the upper bound and min-error as the lower bound for the stopping conditions.

2 Related Work

The ability of the active learner can be referred to as *selective sampling*, of which two major schemes exist: *uncertainty sampling* and *committee-based sampling*. The former method, for example proposed by Lewis and Gale (1994), is to use only one classifier to identify unlabeled examples on which the classifier is least confident. The latter method (McCallum and Nigam, 1998) generates a committee of classifiers (always more than two classifiers) and selects the next unlabeled example by the principle of maximal disagreement among these classifiers. With selective sampling, the size of the training data can be significantly reduced for text classification (Lewis and Gale, 1994; McCallum and Nigam, 1998), and word sense disambiguation (Chen, *et al.* 2006).

A method similar to committee-based sampling is *co-testing* proposed by Muslea et al. (2000), which trains two learners individually on two compatible and uncorrelated views that should be able to reach the same classification accuracy. In practice, however, these conditions of view selection are difficult to meet in real-world word sense disambiguation tasks.

Recently, much work has been done on the class imbalance problem. The well-known approach is *resampling*, in which some training material is duplicated. Two types of popular resampling methods exist for addressing the class imbalance problem: *over-sampling* and *under-sampling*. The basic idea of resampling methods is to change the training data distribution and make the data more balanced. It works ok in supervised learning, but has not been tested in active learning. Previous work reports that cost-sensitive learning is a good solution to the class imbalance problem (Weiss, 2004). In practice, for WSD, the costs of various senses of a disambiguated word are unequal and unknown, and they are difficult to evaluate in the process of learning.

In recent years, there have been attempts to apply active learning for word sense disambiguation (Chen *et al.*, 2006). However, to our best knowledge, there has been no such attempt to consider the class imbalance problem in the process of active learning for WSD tasks.

3 Resampling Methods

3.1 Under-sampling

Under-sampling is a popular method in addressing the class imbalance problem by changing the training data distribution by removing some exemplars of the majority class at random. Some previous work reported that under-sampling is effective in learning on large imbalanced data sets (Japkowicz and Stephen, 2002). However, as under-sampling removes some potentially useful training samples, it could cause negative effects on the classifier performance.

One-sided sampling is a method similar to under-sampling, in which redundant and borderline training examples are identified and removed from training data (Kubat and Matwin, 1997). Kuban and Matwin reported that one-sided sampling is effective in learning with two-class large imbalanced data sets. However, the relative computational cost of one-sided sampling in active learning is very high, because sampling computations must be implemented for each learning iteration. Our primitive experimental results show that, in the multi-class problem of WSD, one-sided sampling degrades the performance of active learning. And due to the high computation complexity of one-sided sampling, we use random under-sampling in our comparison experiments instead.

To control the degree of change of the training data distribution, the ratio of examples from the majority and the minority class after removal from the majority class is called the *removal rate* (Jo and Japkowicz, 2004). If the removal rate is 1.0, then under-sampling methods build data sets with complete class balance. However, it was reported previously that perfect balance is not always the optimal rate (Estabrooks *et al.*, 2004). In our comparison experiments, we set the removal rate for under-sampling to 0.8, since some cases have 0.8 as the optimal rate reported in (Estabrooks *et al.*, 2004).

3.2 Over-sampling

Over-sampling is also a popular method in addressing the class imbalance problem by resampling the small class until it contains as many examples as the large one. In contrast to under-sampling, over-sampling is the process of adding examples to the minority class, and is accomplished by random sampling and duplication. Because the process of over-sampling involves making exact copies of examples, it usually increases the training cost and may lead to overfitting. There is a recent variant of over-sampling named SMOTE (Chawla *et al.*, 2002) which is a synthetic minority over-sampling technique. The authors reported that a combination of SMOTE and under-sampling can achieve better classifier performance in ROC space than only under-sampling the majority class.

In our comparison experiments, we use over-sampling, measured by a resampling rate called the *addition rate* (Jo and Japkowicz, 2004) that indicates the number of examples that should be added into the minority class. The addition rate for over-sampling is also set to 0.8 in our experiments.

3.3 Bootstrap-based Over-sampling

While over-sampling decreases the between-class imbalance, it increases the within-class imbalance (Jo and Japkowicz, 2004) because of the increase of exact copies of examples at random. To alleviate this within-class imbalance problem, we propose a *bootstrap-based over-sampling* method (BootOS) that uses a bootstrap resampling technique in the process of over-sampling. Bootstrapping, explained below, is a resampling technique similar to *jackknifing*.

There are two reasons for choosing a bootstrap method as resampling technique in the process of over-sampling. First, using a bootstrap set can avoid exactly copying samples in the minority class. Second, the bootstrap method may give a smoothing of the distribution of the training samples (Hamamoto *et al.*, 1997), which can alleviate the within-class imbalance problem cased by over-sampling.

To generate the bootstrap set, we use a well-known bootstrap technique proposed by Hamamoto *et al.* (1997) that does not select samples randomly, allowing all samples in the minority class(es) an equal chance to be selected.

```
Algorithm BootOS(X, N, r, k)
Input: Minority class sample set X={$x_1$, $x_2$, …, $x_n$} of size n; Difference in number of examples between the majority and the minority class = N; Addition rate = r (< 1.0); Number of nearest neighbors = k.
Output: bootstrap sample set $X_B$ of size N*r =X ∪ ($x_{B1}$, $x_{B2}$, …, $x_{B(N*r)}$).
For i = 1 To N*r
    If i == n then (*all samples in minority class sample set have been used*)
        j = 1; //the first sample is selected again
    Else
        j = i; // the i-th sample is selected
    Endif
    Select j-th sample $x_j$ (also as $x_{j,0}$) from X
    Find the k nearest neighbor samples $x_{j,1}$, $x_{j,2}$, …, $x_{j,k}$ using similarity functions.
    Compute a bootstrap sample $x_{Bi}$:
        $x_{Bi} = \frac{1}{k+1}\sum_{l=0}^{k} x_{j,l}$
Endfor
return
```

Figure 1. The BootOS algorithm

4 Active Learning with Resampling

In this work, we are interested in selective sampling for pool-based active learning, and focus on uncertainty sampling (Lewis and Gale, 1994). The key point is how to measure the uncertainty of an unlabeled exemplar, and select a new exemplar with maximum uncertainty to augment the training data. The maximum uncertainty implies that the current classifier has the least confidence in its classification of this exemplar. The well-known *entropy* is a good uncertainty measurement widely used in active learning (zhang and Chen, 2002; Chen *et al.*, 2006):

$$U(i) = H(P_i) = -\sum_{j=1}^{n_i} p(s_j \mid w_i) \log p(s_j \mid w_i) \quad (1)$$

where U is the uncertainty measurement function H represents the entropy function. In the WSD task, $p(s_j|w_i)$ is the predicted probability of sense s_j outputted by the current classifier, when given a sample i containing a disambiguated word w_i.

```
Algorithm Active-Learning-with-Resampling(L,U,m)
Input: Let L be initial small training data set; U the
pool of unlabeled exemplars
Output: labeled training data set L
1.  Resample L to generate new training data set L*
    using resampling techniques such as under-
    sampling, over-sampling or BootOS, and then use
    L* to train the initial classifier
2.  Loop while adding new instances into L
    a.  use the current classifier to probabilistically la-
        bel all unlabeled exemplars in U
    b.  Based on active learning rules, present m top-
        ranked exemplars to oracle for labeling
    c.  Augment L with the m new exemplars, and re-
        move them from U
    d.  Resample L to generate new training data set
        L* using resampling techniques such as under-
        sampling, over-sampling, or BootOS, and use
        L* to retrain the current classifier
3.  Until the predefined stopping condition is met.
4.  return
```

Figure 2. Active learning with resampling

In step 1 and 2(d) in Fig. 2, if we do not generate L*, and L is used directly to train the current classifier, we call it *ordinary active learning*. In the process of active learning, we used the entropy-based uncertainty measurement for all active learning frameworks in our comparison experiments. Actually our active learning with resampling is a heterogeneous approach in which the classifier used to select new instances is different from the resulting classifier (Lewis and Catlett, 1994).

We utilize a maximum entropy (ME) model (Berger *et al.*, 1996) to design the basic classifier used in active learning for WSD. The advantage of the ME model is the ability to freely incorporate features from diverse sources into a single, well-grounded statistical model. A publicly available ME toolkit (Zhang *et. al.*, 2004) was used in our experiments. In order to extract the linguistic features necessary for the ME model, all sentences containing the target word were automatically part-of-speech (POS) tagged using the Brill POS tagger (Brill, 1992). Three knowledge sources were used to capture contextual information: unordered single words in topical context, POS of neighboring words with position information, and local collocations. These are same as three of the four knowledge sources used in (Lee and Ng, 2002). Their fourth knowledge source (named syntactic relations) was not used in our work.

5 Stopping Conditions

In active learning algorithm, defining the stopping condition for active learning is a critical problem, because it is almost impossible for the human annotator to label all unlabeled samples. This is a problem of estimation of classifier effectiveness (Lewis and Gale 1994). In fact, it is difficult to know when the classifier reaches maximum effectiveness. In previous work some researchers used a simple stopping condition when the training set reached a predefined desired size. It is almost impossible to predefine an appropriate size of desirable training data for inducing the most effective classifier.

To solve the problem, we consider the problem of estimating classifier effectiveness as the problem of confidence estimation of classifier on the remaining unlabeled samples. Concretely, if we find that the current classifier already has acceptably strong confidence on its classification results for all remained unlabeled data, we assume the current training data is sufficient to train the classifier with maximum effectiveness. In other words, if a classifier induced from the current training data has strong classification confidence on an unlabeled example, we could consider it as a redundant example.

Based on above analyses, we adopt here two stopping conditions for active learning:

- ***Max-confidence*:** This strategy is based on uncertainty measurement, considering whether the entropy of each selected unlabeled example is less than a very small predefined threshold close to zero, such as 0.001.
- ***Min-error***: This strategy is based on feedback from the *oracle* when the active learner asks for true labels for selected unlabeled examples, considering whether the current trained classifier could correctly predict the labels or the accuracy performance of predictions on

selected unlabeled examples is already larger than a predefined accuacy threshold.

Once *max-confidence* and *min-error* conditions are met, the current classifier is assumed to have strong enough confidence on the classification results of all remained unlabeled data.

6 Evaluation

6.1 Data

The data used for our comparison experiments were developed as part of the OntoNotes project (Hovy *et al.*, 2006), which uses the WSJ part of the Penn Treebank (Marcus *et al.*, 1993). The senses of noun words occurring in OntoNotes are linked to the Omega ontology. In OntoNotes, at least two humans manually annotate the coarse-grained senses of selected nouns and verbs in their natural sentence context. To date, OntoNotes has annotated several tens of thousands of examples, covering several hundred nouns and verbs, with an inter-annotator agreement rate of at least 90%.

Those 38 random chosen ambiguous nouns used in all following experiments are shown in Table 1. It is apparent that the sense distributions of most nouns are very skewed (frequencies shown in the table, separated by /).

Words	sense distribution	words	sense distribution
Rate	1025/182	president	936/157/17
People	815/67/7/5	part	456/102/75/16
Point	471/88/37/19/9/6	director	517/23
Revenue	517/23	bill	348/130/40
Future	413/82/23	order	354/61/54/6/6
Plant	376/51	board	369/15
Today	238/149	policy	308/74
Capital	325/21/8	term	147/137/52/13
management	210/130	move	302/13/5
Position	97/75/67/61/10/7	amount	236/57/16
Home	267/17/16	power	154/134/15
Leader	244/38	return	191/35/29/12/9
administration	266/11	payment	201/69
Account	233/18/13	control	90/66/64/21/12/5
Lot	221/20	activity	218/23
Drug	160/74	building	177/48/5
Estate	214/11	house	112/71/25
development	165/46/6	network	127/53/29
Strategy	198/11	place	69/63/50/18/5

Table 1. Data set used in experiments

6.2 Results

In the following active learning comparison experiments, we tested with five resampling methods including *random sampling* (Random), *uncertainty sampling* (Ordinary), *under-sampling*, *over-sampling*, and *BootOS*. The 1-NN technique was used for bootstrap-based resampling of BootOS in our experiments. A 5 by 5-fold cross-validation was performed on each noun's data.

We used 20% randomly chosen data for held-out evaluation and the other 80% as the pool of unlabeled data for each round of the active learning. For all words, we started with a randomly chosen initial training set of 10 examples, and we made 10 queries after each learning iteration.

In the evaluation, average *accuracy* and *recall* are used as measures of performances for each active learning method. Note that the *macro-average* way is adopted for recall evaluation in each noun WSD task. The accuracy measure indicates the percentage of testing instances correctly identified by the system. The macro-average recall measure indicates how well the system performs on each sense.

Experiment 1: Performance comparison experiments on active learning

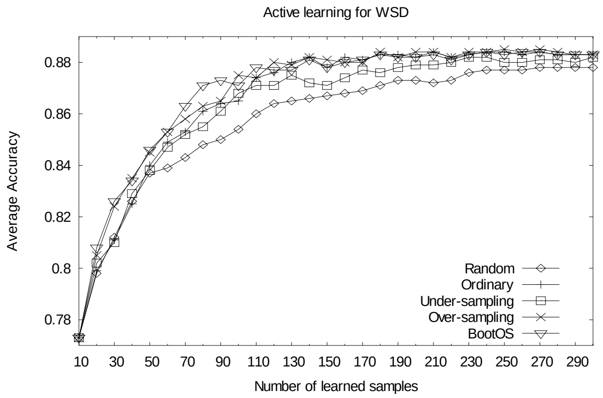


Figure 3. Average accuracy performance comparison experiments

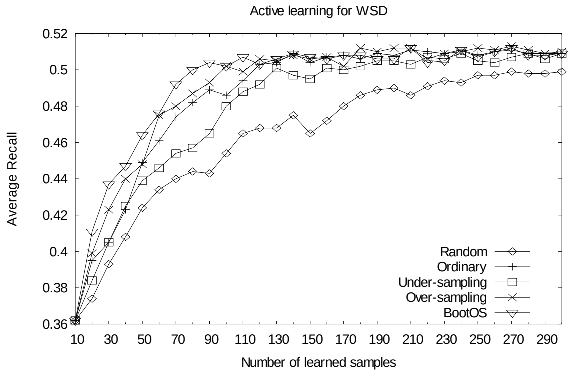


Figure 4. Average recall performance comparison experiments

As shown in Fig. 3 and Fig. 4, when the number of learned samples for each noun is smaller than 120, the BootOS has the best performance, followed by over-sampling and ordinary method. As the number of learned samples increases, ordinary, over-sampling and BootOS have similar performances on accuracy and recall. Our experiments also exhibit that random sampling method is the worst on both accuracy and recall.

Previous work (Estabrooks *et al.*, 2004) reported that under-sampling of the majority class (predominant sense) has been proposed as a good means of increasing the sensitivity of a classifier to the minority class (infrequent sense). However, in our active learning experiments, under-sampling is apparently worse than ordinary, over-sampling and our BootOS. The reason is that in highly imbalanced data, too many useful training samples of majority class are discarded in under-sampling, causing the performance of active learning to degrade.

Experiment 2: Effectiveness of learning instances for infrequent senses

It is important to enrich the corpora by learning more instances for infrequent senses using active learning with less human labeling. This procedure not only makes the corpora 'richer', but also alleviates the domain dependence problem faced by corpus-based supervised approaches to WSD.

The objective of this experiment is to evaluate the performance of active learning in learning samples of infrequent senses from an unlabeled corpus. Due to highly skewed word sense distributions in our data set, we consider all senses other than the predominant sense as infrequent senses in this experiment.

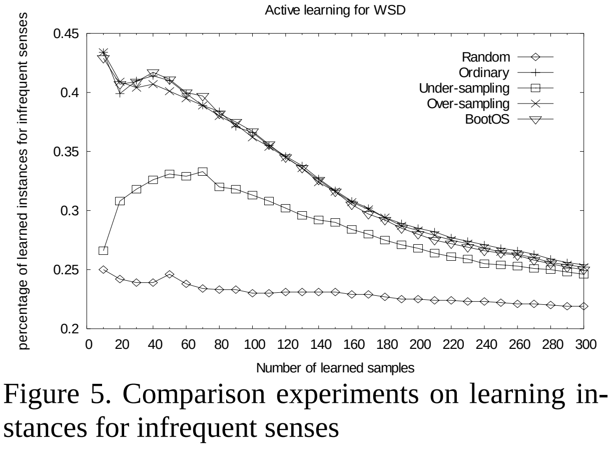


Figure 5. Comparison experiments on learning instances for infrequent senses

Fig. 5 shows that random sampling is the worst in active learning for infrequent senses. The reason is very obvious: the sense distribution of the learned sample set by random sampling is almost identical to that of the original data set.

Under-sampling is apparently worse than ordinary active learning, over-sampling and BootOS methods. When the number of learned samples for each noun is smaller than 80, BootOS achieves slight better performance than ordinary active learning and over-sampling.

When the number of learned samples is larger than 80 and smaller than 160, these three methods exhibit similar performance. As the number of iterations increases, ordinary active learning is slightly better than over-sampling and BootOS. In fact, after the 16th iteration (10 samples chosen in each iteration), results indicate that most instances for infrequent senses have been learned.

Experiment 3: Effectiveness of Stopping Conditions for active learning

To evaluate the effectiveness of two strategies *max-confidence* and *min-error* as stopping conditions of active learning, we first construct an ideal stopping condition when the classifier could reach the highest accuracy performance at the first time in the procedure of active learning. When the ideal stopping condition is met, it means that the current classifier has reached maximum effectiveness. In practice, it is impossible to exactly know when the ideal stopping condition is met before all unlabeled data are labeled by a human annotator. We only use this ideal method in our comparison experiments to analyze the effectiveness of our two proposed stopping conditions.

For general purpose, we focus on the ordinary active learning to design the basic system, and to evaluate the effectiveness of three stop conditions. In the following experiments, the entropy threshold used in *max-confidence* strategy is set to 0.001, and the accuracy threshold used in *min-error strategy* is set to 0.9.

In Table 2, the column "*Size*" stands for the size of unlabeled data set of corresponding noun word used in active learning. There are two columns for each stopping condition: the left column "*num*" presents number of learned instances and the right column "*%*" presents its percentage over all data when the corresponding stopping condition is met.

Words	Size	Ideal		Max-confidence		Min-error	
		num	%	num	%	num	%
Rate	966	200	.23	410	.41	290	.29
People	715	140	.20	290	.41	200	.28
Point	504	90	.18	220	.44	120	.24
Revenue	432	70	.16	110	.25	80	.19
Future	414	120	.29	140	.34	60	.14
Plant	342	210	.61	180	.53	110	.32
Today	382	250	.65	240	.63	230	.60
Capital	283	70	.25	180	.64	90	.32
Management	272	200	.74	210	.77	210	.77
Position	254	210	.83	230	.91	220	.87
Home	240	60	.25	160	.67	60	.25
Leader	226	60	.27	120	.53	70	.31
administration	222	30	.14	90	.41	50	.23
Account	211	50	.24	130	.62	70	.33
Lot	185	30	.16	60	.32	40	.22
Drug	187	130	.70	140	.75	120	.64
Estate	180	20	.11	50	.28	30	.17
Development	174	40	.23	150	.86	80	.46
Strategy	167	10	.06	100	.60	10	.06
President	888	120	.14	220	.25	120	.14
Part	519	110	.21	240	.46	130	.25
Director	432	110	.25	130	.30	90	.21
Bill	414	120	.29	280	.68	150	.36
Order	385	130	.34	220	.57	140	.36
Board	307	40	.13	190	.62	40	.13
Policy	306	90	.29	200	.65	150	.49
Term	279	120	.43	190	.68	130	.47
Move	256	50	.20	140	.55	50	.20
Amount	247	210	.85	200	.81	140	.57
Power	242	190	.78	190	.78	190	.78
Return	221	90	.41	160	.72	100	.45
Payment	216	120	.56	160	.74	150	.69
Control	206	160	.78	200	.97	200	.97
Activity	193	30	.16	130	.67	70	.36
Building	184	90	.49	130	.71	110	.60
House	166	100	.60	150	.90	110	.66
Network	167	110	.66	130	.78	100	.60
Place	164	120	.73	150	.91	120	.73

Table 2 Effectiveness of three stopping conditions

As shown in Table 2, the min-error strategy based on feedback of human annotator is very close to the ideal method. Therefore, when comparing to ideal stopping condition, min-error strategy is a good choice as stopping condition for active learning. It is important to note that the min-error method does not need more additional computational costs, it only depends upon the feedback of human annotator when labeling the chosen unlabeled samples.

From experimental results, we can see that max-confidence strategy is worse than min-error method. However, we believe that the entropy of each unlabeled sample is a good signal to stop active learning. So we suggest that there may be a good prediction solution in which the min-error strategy is used as the lower-bound of stopping condition, and max-confidence strategy as the upper-bound of stopping condition for active learning.

7 Discussion

As discussed above, finding more instances for infrequent senses at the earlier stages of active learning is very significant in making the corpus richer, meaning less effort for human labeling. In practice, another way to learn more instances for infrequent senses is to first build a training data set by active learning or by human efforts, and then build a supervised classifier to find more instances for infrequent sense. However, it is interesting to know how much initial training data is enough for this task, and how much human labeling efforts could be saved.

From experimental results, we found that among these chosen unlabeled instances by active learner, some instances are informative samples helpful for improving classification performance, and other instances are borderline samples which are unreliable because even a small amount of noise can lead the sample to the wrong side of the decision boundary. The removal of these borderline samples might improve the performance of active learning.

The proposed prediction solution based on max-confidence and min-error strategies is a coarse framework. To predict when to stop active learning procedure, it is logical to consider the changes of accuracy performance of the classifier as a signal to stop the learning iteration. In other words, during the range predicted by the proposed solution, if the change of accuracy performance of the learner (classifier) is very small, we could assume that the current classifier has reached maximum effectiveness.

8 Conclusion and Future Work

In this paper, we consider the class imbalance problem in WSD tasks, and analyze the effect of resampling techniques including over-sampling and under-sampling in active learning. Experimental results show that over-sampling is a relatively good choice in active learning for WSD in highly imbalanced data. Under-sampling causes negative effect on active learning. A new over-sampling method named BootOS based on bootstrap technique is proposed to alleviate the within-class imbalance problem of over-sampling, and works better than ordinary over-sampling in active learning for WSD. It is noteworthy that none of these techniques require to modify the architecture or learning algorithm; therefore, they are very easy to use and extend to other applications. To predict when to stop active learning, we adopt two strategies including max-confidence and min-error as stopping conditions. According to our experimental results, we suggest a prediction solution by considering max-confidence as the upper bound and min-error as the lower bound of stopping conditions for active learning.

In the future work, we will study how to exactly identify these borderline samples thus they are not firstly selected in active learning procedure. The borderline samples have the higher entropy values meaning least confident for the current classifier. The borderline instances can be detected using the concept of *Tomek links* (Tomek 1976). It is also worth studying cost-sensitive learning for active learning with imbalanced data, and using such techniques for WSD.

References

A. L. Berger, S. A. Della, and V. J Della. 1996. *A maximum entropy approach to natural language processing*. Computational Linguistics 22(1):39–71.

E Brill. 1992. *A simple rule-based part of speech tagger*. In the Proceedings of the Third Conference on Applied Natural Language Processing.

Y. S. Chan and H. T. Ng. 2006. *Estimating class priors in domain adaptation*. In Proc. of ACL06.

N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer. 2002. *SMOTE: synthetic minority over-sampling technique*. Journal of Artificial Intelligence Research, 2002(16): 321-357

J. Chen, A. Schein, L. Ungar, M. Palmer. 2006. *An empirical study of the behavior of active learning for word sense disambiguation*. In Proc. of HLT-NAACL06

I. Dagan, O. Glickman, A. Gliozzo, E. Marmorshtein, and C. Strapparava. 2006. *Direct Word Sense Matching for Lexical Substitution*. In Proc. of ACL'06

H. T. Dang and M. Palmer. 2005. *The Role of Semantic Roles in Disambiguating Verb Senses*. In Proc. of ACL'05.

A. Estabrooks, T. Jo and N. Japkowicz. 2004. *A multiple resampling method for learning from imbalanced data set*. Computational Intelligence, 20(1):18-36

Y. Hamamoto, S. Uchimura and S. Tomita. 1997. *A bootstrap technique for nearest neighbor classifier design*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(1):73-79

V. Hoste, A. Kool, and W. Daelemans. 2001. *Classifier optimization and combination in the English all words task*. In Proc. of the SENSEVAL-2 workshop

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw and R. Weischedel. 2006. *Ontonotes: The 90% Solution*. In Proc. of HLT-NAACL06.

N. Ide and J. Veronis. 1998. *Introduction to the special issue on word sense disambiguation: the state of the art*. Computational Linguistics, 24(1):1-37

N. Japkowicz and S. Stephen. 2002. *The class imbalance problem: a systematic study*. Intelligent Data Analysis, 6(5):429-450

T. Jo and N. Japkowicz. 2004. *Class imbalances versus small disjuncts*. SIGKSS Explorations, 6(1):40-49

U. S. Kohomban and W. S. Lee. 2005. *Learning Semantic Classes for Word Sense Disambiguation*. In Proc. of ACL'05

M. Kubat and S. Matwin. 1997. *Addressing the curse of imbalanced training sets: one-sided selection*. In Proc. of ICML97

Y.K. Lee and. H.T. Ng. 2002. *An empirical evaluation of knowledge sources and learning algorithm for word sense disambiguation*. In Proc. of EMNLP-2002

D. D. Lewis and W. A. Gale. 1994. *A sequential algorithm for training text classifiers*. In Proc. of SIGIR-94

D.D. Lewis and J. Catlett. 1994. *Heterogeneous uncertainty sampling for supervised learning*. In Proc. of ICML94

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. *Building a large annotated corpus of English: the Penn Treebank*. Computational Linguistics, 19(2):313-330

A. McCallum and K. Nigram. 1998. *Employing EM in pool-based active learning for text classification*. In Proc. 15th ICML

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. 2004. *Finding predominant senses in untagged text*. In Proc. of ACL04

I. Muslea, S. Minton, and C. A. Knoblock. 2000. *Selective sampling with redundant views*. In Proc. of National Conference on Artificial Intelligence

I. Tomek. 1976. *Two modifications of CNN*. IEEE Transactions on Systems, Man and Cybernetics, 6(6):769-772

G. M. Weiss. 2004. *Mining with rarity – problems and solutions: a unifying framework*. SIGKDD Explorations, 6(1):7-19

N. Xue, J. Chen and M. Palmer. 2006. *Aligning Features with Sense Distinction Dimensions*. In Proc. of ACL'06

Z. Zhou, X. Liu. 2006. *Training cost-sensitive neural networks with methods addressing the class imbalance problem*. IEEE Transactions on Knowledge and Data Engineering, 18(1):63-77

L. Zhang, J. Zhu, and T. Yao. 2004. *An evaluation of statistical spam filtering techniques*. ACM Transactions on Asian Language Information Processing, 3(4):243–269.

C. Zhang and T. Chen. 2002. *An active learning framework for content-based information retrieval*. IEEE Transactions on Multimedia, 4(2):260-268

Semi-Supervised Structured Output Learning based on a Hybrid Generative and Discriminative Approach

Jun Suzuki, Akinori Fujino and Hideki Isozaki
NTT Communication Science Laboratories, NTT Corp.
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{jun, a.fujino, isozaki}@cslab.kecl.ntt.co.jp

Abstract

This paper proposes a framework for semi-supervised structured output learning (SOL), specifically for sequence labeling, based on a hybrid generative and discriminative approach. We define the objective function of our hybrid model, which is written in log-linear form, by discriminatively combining discriminative structured predictor(s) with generative model(s) that incorporate unlabeled data. Then, unlabeled data is used in a generative manner to increase the sum of the discriminant functions for all outputs during the parameter estimation. Experiments on named entity recognition (CoNLL-2003) and syntactic chunking (CoNLL-2000) data show that our hybrid model significantly outperforms the state-of-the-art performance obtained with supervised SOL methods, such as conditional random fields (CRFs).

1 Introduction

Structured output learning (SOL) methods, which attempt to optimize an interdependent output space globally, are important methodologies for certain natural language processing (NLP) tasks such as part-of-speech tagging, syntactic chunking (Chunking) and named entity recognition (NER), which are also referred to as sequence labeling tasks. When we consider the nature of these sequence labeling tasks, a semi-supervised approach appears to be more natural and appropriate. This is because the number of features and parameters typically become extremely large, and labeled examples can only sparsely cover the parameter space, even if thousands of labeled examples are available. In fact, many attempts have recently been made to develop semi-supervised SOL methods (Zhu et al., 2003; Li and McCallum, 2005; Altun et al., 2005; Jiao et al., 2006; Brefeld and Scheffer, 2006).

With the generative approach, we can easily incorporate unlabeled data into probabilistic models with the help of expectation-maximization (EM) algorithms (Dempster et al., 1977). For example, the Baum-Welch algorithm is a well-known algorithm for training a hidden Markov model (HMM) of sequence learning. Generally, with sequence learning tasks such as NER and Chunking, we cannot expect to obtain better performance than that obtained using discriminative approaches in supervised learning settings.

In contrast to the generative approach, with the discriminative approach, it is not obvious how unlabeled training data can be naturally incorporated into a discriminative training criterion. For example, the effect of unlabeled data will be eliminated from the objective function if the unlabeled data is directly used in traditional i.i.d. conditional-probability models. Nevertheless, several attempts have recently been made to incorporate unlabeled data in the discriminative approach. An approach based on pairwise similarities, which encourage nearby data points to have the same class label, has been proposed as a way of incorporating unlabeled data discriminatively (Zhu et al., 2003; Altun et al., 2005; Brefeld and Scheffer, 2006). However, this approach generally requires joint inference over the whole data set for prediction, which is not practical as regards the large data sets used for standard sequence labeling tasks in NLP. Another discriminative approach to semi-supervised SOL involves the incorporation of an entropy regularizer (Grand-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 791–800, Prague, June 2007. ©2007 Association for Computational Linguistics

valet and Bengio, 2004). Semi-supervised conditional random fields (CRFs) based on a minimum entropy regularizer (SS-CRF-MER) have been proposed in (Jiao et al., 2006). With this approach, the parameter is estimated to maximize the likelihood of labeled data and the negative conditional entropy of unlabeled data. Therefore, the structured predictor is trained to separate unlabeled data well under the entropy criterion by parameter estimation.

In contrast to these previous studies, this paper proposes a semi-supervised SOL framework based on a hybrid generative and discriminative approach. A hybrid approach was first proposed in a supervised learning setting (Raina et al., 2003) for text classification. (Fujino et al., 2005) have developed a semi-supervised approach by discriminatively combining a supervised classifier with generative models that incorporate unlabeled data. We extend this framework to the structured output domain, specifically for sequence labeling tasks. Moreover, we reformalize the objective function to allow the incorporation of discriminative models (structured predictors) trained from labeled data, since the original framework only considers the combination of generative classifiers. As a result, our hybrid model can significantly improve on the state-of-the-art performance obtained with supervised SOL methods, such as CRFs, even if a large amount of labeled data is available, as shown in our experiments on CoNLL-2003 NER and CoNLL-2000 Chunking data. In addition, compared with SS-CRF-MER, our hybrid model has several good characteristics including a low calculation cost and a robust optimization in terms of a sensitiveness of hyper-parameters. This is described in detail in Section 5.3.

2 Supervised SOL: CRFs

This paper focuses solely on sequence labeling tasks, such as named entity recognition (NER) and syntactic chunking (Chunking), as SOL problems. Thus, let $\boldsymbol{x}{=}(x_1,\dots,x_S)\in\mathcal{X}$ be an input sequence, and $\boldsymbol{y}{=}(y_0,\dots,y_{S+1})\in\mathcal{Y}$ be a particular output sequence, where y_0 and y_{S+1} are special fixed labels that represent the beginning and end of a sequence.

As regards supervised sequence learning, CRFs are recently introduced methods that constitute flexible and powerful models for structured predictors based on undirected graphical models that have been globally conditioned on a set of inputs (Lafferty et al., 2001). Let $\boldsymbol{\lambda}$ be a parameter vector and $\boldsymbol{f}(y_{s-1},y_s,\boldsymbol{x})$ be a (local) feature vector obtained from the corresponding position s given $\boldsymbol{x}$. CRFs define the conditional probability, $p(\boldsymbol{y}|\boldsymbol{x})$, as being proportional to a product of potential functions on the cliques. That is, $p(\boldsymbol{y}|\boldsymbol{x})$ on a (linear-chain) CRF can be defined as follows:

$$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})=\frac{1}{Z(\boldsymbol{x})}\prod_{s=1}^{S+1}\exp(\boldsymbol{\lambda}\cdot\boldsymbol{f}(y_{s-1},y_s,\boldsymbol{x})).$$

$Z(\boldsymbol{x})=\sum_{\boldsymbol{y}}\prod_{s=1}^{S+1}\exp(\boldsymbol{\lambda}\cdot\boldsymbol{f}(y_{s-1},y_s,\boldsymbol{x}))$ is a normalization factor over all output values, $\mathcal{Y}$, and is also known as the partition function.

For parameter estimation (training), given labeled data $\mathcal{D}_l=\{(\boldsymbol{x}^k,\boldsymbol{y}^k)\}_{k=1}^K$, the Maximum a Posteriori (MAP) parameter estimation, namely maximizing $\log p(\boldsymbol{\lambda}|\mathcal{D}_l)$, is now the most widely used CRF training criterion. Thus, we maximize the following objective function to obtain optimal $\boldsymbol{\lambda}$:

$$\mathcal{L}^{\text{CRF}}(\boldsymbol{\lambda})=\sum_k\Big[\boldsymbol{\lambda}\cdot\sum_s\boldsymbol{f}_s-\log Z(\boldsymbol{x}^k)\Big]+\log p(\boldsymbol{\lambda}),\quad(1)$$

where $\boldsymbol{f}_s$ is an abbreviation of $\boldsymbol{f}(y_{s-1},y_s,\boldsymbol{x})$ and $p(\boldsymbol{\lambda})$ is a prior probability distribution of $\boldsymbol{\lambda}$. A gradient-based optimization algorithm such as L-BFGS (Liu and Nocedal, 1989) is widely used for maximizing Equation (1). The gradient of Equation (1) can be written as follows:

$$\begin{aligned}\nabla\mathcal{L}^{\text{CRF}}(\boldsymbol{\lambda})=&\sum_k E_{\tilde{p}(\boldsymbol{y}^k,\boldsymbol{x}^k;\boldsymbol{\lambda})}\big[\sum_s\boldsymbol{f}_s\big]\\&-\sum_k E_{p(\mathcal{Y}|\boldsymbol{x}^k;\boldsymbol{\lambda})}\big[\sum_s\boldsymbol{f}_s\big]+\nabla\log p(\boldsymbol{\lambda}).\end{aligned}$$

Calculating $E_{p(\mathcal{Y}|\boldsymbol{x},\boldsymbol{\lambda})}$ as well as the partition function $Z(\boldsymbol{x})$ is not always tractable. However, for linear-chain CRFs, a dynamic programming algorithm similar in nature to the forward-backward algorithm in HMMs has already been developed for an efficient calculation (Lafferty et al., 2001).

For prediction, the most probable output, that is, $\hat{\boldsymbol{y}}=\arg\max_{\boldsymbol{y}\in\mathcal{Y}}p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$, can be efficiently obtained by using the Viterbi algorithm.

3 Hybrid Generative and Discriminative Approach to Semi-Supervised SOL

In this section, we describe our formulation of a hybrid approach to SOL and a parameter estimation method for sequence predictors. We assume that we have a set of labeled and unlabeled data, $\mathcal{D} = \{\mathcal{D}_l, \mathcal{D}_u\}$, where $\mathcal{D}_l = \{(\boldsymbol{x}^n, \boldsymbol{y}^n)\}_{n=1}^N$ and $\mathcal{D}_u = \{\boldsymbol{x}^m\}_{m=1}^M$.

Let us assume that we have I-units of discriminative models, p_i^D, and J-units of generative models, p_j^G. Our hybrid model for a structured predictor is designed by the discriminative combination of several joint probability densities of $\boldsymbol{x}$ and $\boldsymbol{y}$, $p(\boldsymbol{x}, \boldsymbol{y})$. That is, the posterior probability of our hybrid model is defined by providing the log-values of $p(\boldsymbol{x}, \boldsymbol{y})$ as the features of a log-linear model, such that:

$$\begin{aligned} &R(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\Lambda}, \boldsymbol{\Theta}, \boldsymbol{\Gamma}) \\ &\quad = \frac{\prod_i p_i^D(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\lambda}_i)^{\gamma_i} \prod_j p_j^G(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_j)^{\gamma_j}}{\sum_{\boldsymbol{y}} \prod_i p_i^D(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\lambda}_i)^{\gamma_i} \prod_j p_j^G(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_j)^{\gamma_j}} \\ &\quad = \frac{\prod_i p_i^D(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\lambda}_i)^{\gamma_i} \prod_j p_j^G(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_j)^{\gamma_j}}{\sum_{\boldsymbol{y}} \prod_i p_i^D(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\lambda}_i)^{\gamma_i} \prod_j p_j^G(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_j)^{\gamma_j}}. \end{aligned} \tag{2}$$

Here, $\boldsymbol{\Gamma} = \{\{\gamma_i\}_{i=1}^I, \{\gamma_j\}_{j=I+1}^{I+J}\}$ represents the discriminative combination weight of each model where $\gamma_i, \gamma_j \in [0, 1]$. Moreover, $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_i\}_{i=1}^I$ and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_j\}_{j=1}^J$ represent model parameters of individual models estimated from labeled and unlabeled data, respectively. Using $p^D(\boldsymbol{x}, \boldsymbol{y}) = p^D(\boldsymbol{y}|\boldsymbol{x}) p^D(\boldsymbol{x})$, we can derive the third line from the second line, where $p_i^D(\boldsymbol{x}; \boldsymbol{\lambda}_i)^{\gamma_i}$ for all i are canceled out. Thus, our hybrid model is constructed by combining discriminative models, $p_i^D(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\lambda}_i)$, with generative models, $p_j^G(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_j)$.

Hereafter, let us assume that our hybrid model consists of CRFs for discriminative models, p_i^D, and HMMs for generative models, p_j^G, shown in Equation (2), since this paper focuses solely on sequence modeling. For HMMs, we consider a first order HMM defined in the following equation:

$$p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) = \prod_{s=1}^{S+1} \theta_{y_{s-1}, y_s} \theta_{y_s, x_s},$$

where θ_{y_{s-1}, y_s} and θ_{y_s, x_s} represent the transition probability between states y_{s-1} and y_s and the symbol emission probability of the s-th position of the corresponding input sequence, respectively, where $\theta_{y_{S+1}, x_{S+1}} = 1$.

It can be seen that the formalization in the log-linear combination of our hybrid model is very similar to that of LOP-CRFs (Smith et al., 2005). In fact, if we only use a combination of discriminative models (CRFs), which is equivalent to $\gamma_j = 0$ for all j, we obtain essentially the same objective function as that of the LOP-CRFs. Thus, our framework can also be seen as an extension of LOP-CRFs that enables us to incorporate unlabeled data.

3.1 Discriminative Combination

For estimating the parameter $\boldsymbol{\Gamma}$, let us assume that we already have discriminatively trained models on labeled data, $p_i^D(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\lambda}_i)$. We maximize the following objective function for estimating parameter $\boldsymbol{\Gamma}$ under a fixed $\boldsymbol{\Theta}$:

$$\mathcal{L}^{\text{HySOL}}(\boldsymbol{\Gamma}|\boldsymbol{\Theta}) = \sum_n \log R(\boldsymbol{y}^n|\boldsymbol{x}^n; \boldsymbol{\Lambda}, \boldsymbol{\Theta}, \boldsymbol{\Gamma}) + \log p(\boldsymbol{\Gamma}). \tag{3}$$

where $p(\boldsymbol{\Gamma})$ is a prior probability distribution of $\boldsymbol{\Gamma}$.

The value of $\boldsymbol{\Gamma}$ providing a global maximum of $\mathcal{L}^{\text{HySOL}}(\boldsymbol{\Gamma}|\boldsymbol{\Theta})$ is guaranteed under an arbitrary fixed value in the $\boldsymbol{\Theta}$ domain, since $\mathcal{L}^{\text{HySOL}}(\boldsymbol{\Gamma}|\boldsymbol{\Theta})$ is a concave function of $\boldsymbol{\Gamma}$. Thus, we can easily maximize Equation (3) by using a gradient-based optimization algorithm such as (bound constrained) L-BFGS (Liu and Nocedal, 1989).

3.2 Incorporating Unlabeled Data

We cannot directly incorporate unlabeled data for discriminative training such as Equation (3) since the correct outputs $\boldsymbol{y}$ for unlabeled data are unknown. On the other hand, generative approaches can easily deal with unlabeled data as incomplete data (data with missing variable $\boldsymbol{y}$) by using a mixture model. A well-known way to achieve this incorporation is to maximize the log likelihood of unlabeled data with respect to the marginal distribution of generative models as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_m \log \sum_{\boldsymbol{y}} p(\boldsymbol{x}^m, \boldsymbol{y}; \boldsymbol{\theta}).$$

In fact, (Nigam et al., 2000) have reported that using unlabeled data with a mixture model can improve the text classification performance.

According to Bayes' rule, $p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) \propto p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$, the discriminant functions of generative classifiers are provided by generative models $p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$. Therefore, we can regard $\mathcal{L}(\boldsymbol{\theta})$ as the logarithm of the sum of discriminant functions for all missing variables $\boldsymbol{y}$ of unlabeled data. Following this view, we can directly incorporate unlabeled data into our hybrid model by maximizing the discriminant functions g of our hybrid model in the same way as for a mixture model as explained above. Thus, we maximize the following objective function for estimating the model parameters $\boldsymbol{\Theta}$ for generative models of unlabeled data:

$$\mathcal{G}(\boldsymbol{\Theta}|\boldsymbol{\Gamma}) = \sum_m \log \sum_{\boldsymbol{y}} g(\boldsymbol{x}^m, \boldsymbol{y}; \boldsymbol{\Theta}) + \log p(\boldsymbol{\Theta}). \quad (4)$$

where $p(\boldsymbol{\Theta})$ is a prior probability distribution of $\boldsymbol{\Theta}$. Here, the discriminant function g of output $\boldsymbol{y}$ given input $\boldsymbol{x}$ in our hybrid model can be obtained by the numerator on the third line of Equation (2), since the denominator does not affect the determination of $\boldsymbol{y}$, that is,

$$g(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \prod_i p_i^D(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\lambda}_i)^{\gamma_i} \prod_j p_j^G(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_j)^{\gamma_j}.$$

Under a fixed $\boldsymbol{\Gamma}$, we can estimate the local maximum of $\mathcal{G}(\boldsymbol{\Theta}|\boldsymbol{\Gamma})$ around the initialized value of $\boldsymbol{\Theta}$ by an iterative computation such as the EM algorithm (Dempster et al., 1977). Let $\boldsymbol{\Theta}''$ and $\boldsymbol{\Theta}'$ be estimates of $\boldsymbol{\Theta}$ in the next and current steps, respectively. Using Jensen's inequality, $\log a \leq a - 1$, we obtain a Q-function that satisfies the inequality $\mathcal{G}(\boldsymbol{\Theta}''|\boldsymbol{\Gamma})–\mathcal{G}(\boldsymbol{\Theta}'|\boldsymbol{\Gamma}) \geq Q(\boldsymbol{\Theta}'', \boldsymbol{\Theta}'; \boldsymbol{\Gamma})–Q(\boldsymbol{\Theta}', \boldsymbol{\Theta}'; \boldsymbol{\Gamma})$, such that

$$\begin{aligned} & Q(\boldsymbol{\Theta}'', \boldsymbol{\Theta}'; \boldsymbol{\Gamma}) \\ = & \sum_j \gamma_j \sum_m \sum_{\boldsymbol{y}} R(\boldsymbol{y}|\boldsymbol{x}^m; \boldsymbol{\Lambda}, \boldsymbol{\Theta}', \boldsymbol{\Gamma}) \log p_j^G(\boldsymbol{x}^m, \boldsymbol{y}; \boldsymbol{\Theta}'') \\ & + \log p(\boldsymbol{\Theta}''). \end{aligned} \quad (5)$$

Since $Q(\boldsymbol{\Theta}', \boldsymbol{\Theta}'; \boldsymbol{\Gamma})$ is independent of $\boldsymbol{\Theta}''$, we can improve the value of $\mathcal{G}(\boldsymbol{\Theta}|\boldsymbol{\Gamma})$ by computing $\boldsymbol{\Theta}''$ to maximize $Q(\boldsymbol{\Theta}'', \boldsymbol{\Theta}'; \boldsymbol{\Gamma})$. We can obtain a $\boldsymbol{\Theta}$ estimate by iteratively performing this update while $\mathcal{G}(\boldsymbol{\Theta}|\boldsymbol{\Gamma})$ is hill climbing.

As shown in Equation (5), R is used for estimating the parameter $\boldsymbol{\Theta}$. The intuitive effect of maximizing Equation (4) is similar to performing 'soft-clustering'. That is, unlabeled data is clustered with respect to the R distribution, which also includes information about labeled data, under the constraint of generative model structures.

3.3 Parameter Estimation Procedure

According to our definition, the $\boldsymbol{\Theta}$ and $\boldsymbol{\Gamma}$ estimations are mutually dependent. That is, the parameters of the hybrid model, $\boldsymbol{\Gamma}$, should be estimated using Equation (3) with a fixed $\boldsymbol{\Theta}$, while the parameters of the generative models, $\boldsymbol{\Theta}$, should be estimated using Equation (4) with a fixed $\boldsymbol{\Gamma}$. As a solution to our parameter estimation, we search for the $\boldsymbol{\Theta}$ and $\boldsymbol{\Gamma}$ that maximize $\mathcal{L}^{\text{HySOL}}(\boldsymbol{\Gamma}|\boldsymbol{\Theta})$ and $\mathcal{G}(\boldsymbol{\Theta}|\boldsymbol{\Gamma})$ simultaneously. For this search, we compute $\boldsymbol{\Theta}$ and $\boldsymbol{\Gamma}$ by maximizing the objective functions shown in Equations (4) and (3) iteratively and alternately. We summarize the algorithm for estimating these model parameters in Figure 1.

```
1.Given training set: $\mathcal{D}_u = \{\boldsymbol{x}^m\}_{m=1}^M$ and
        $\mathcal{D}_l = \{\mathcal{D}'_l = \{(\boldsymbol{x}^k, \boldsymbol{y}^k)\}_{k=1}^K, \mathcal{D}''_l = \{(\boldsymbol{x}^n, \boldsymbol{y}^n)\}_{n=1}^N\}$
2.Compute $\boldsymbol{\Lambda}$, using $\mathcal{D}'_l$.
3.Initialize $\boldsymbol{\Gamma}^{(0)}$, $\boldsymbol{\Theta}^{(0)}$ and $t \leftarrow 0$.
4.Perform the following until $\frac{|\boldsymbol{\Theta}^{(t+1)} - \boldsymbol{\Theta}^{(t)}|}{|\boldsymbol{\Theta}^{(t)}|} < \epsilon$.
  4.1. Compute $\boldsymbol{\Theta}^{(t+1)}$ to maximize Equation (4)
          under fixed $\boldsymbol{\Gamma}^{(t)}$ and $\boldsymbol{\Lambda}$ using $\mathcal{D}_u$.
  4.2. Compute $\boldsymbol{\Gamma}^{(t+1)}$ to maximize Equation (3)
          under fixed $\boldsymbol{\Theta}^{(t+1)}$ and $\boldsymbol{\Lambda}$ using $\mathcal{D}''_l$.
  4.3. $t \leftarrow t + 1$.
5.Output a structured predictor $R(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Lambda}, \boldsymbol{\Theta}^{(t)}, \boldsymbol{\Gamma}^{(t)})$.
```

Figure 1: Algorithm of learning model parameters used in our hybrid model.

Note that during the $\boldsymbol{\Gamma}$ estimation (procedure 4.2 in Figure 1), $\boldsymbol{\Gamma}$ can be over-fitted to the labeled training data if we use the same labeled training data as used for the $\boldsymbol{\Lambda}$ estimation. There are several possible ways to reduce this over-fit. In this paper, we select one of the simplest; we divide the labeled training data $\mathcal{D}_l$ into two distinct sets $\mathcal{D}'_l$ and $\mathcal{D}''_l$. Then, $\mathcal{D}'_l$ and $\mathcal{D}''_l$ are individually used for estimating $\boldsymbol{\Lambda}$ and $\boldsymbol{\Gamma}$, respectively. In our experiments, we divide the labeled training data $\mathcal{D}_l$ so that 4/5 is used for $\mathcal{D}'_l$ and the remaining 1/5 for $\mathcal{D}''_l$.

3.4 Efficient Parameter Estimation Algorithm

Let $\mathcal{N}_R(\boldsymbol{x})$ represent the denominator of Equation (2), that is the normalization factor of R. We can rearrange Equation (2) as follows:

$$R(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\Lambda}, \boldsymbol{\Theta}, \boldsymbol{\Gamma}) = \frac{\prod_s \prod_i \left[V_{i,s}^D\right]^{\gamma_i} \prod_j \left[V_{j,s}^G\right]^{\gamma_j}}{\mathcal{N}_R(\boldsymbol{x}) \prod_i [Z_i(\boldsymbol{x})]^{\gamma_i}}, \quad (6)$$

where $V_{i,s}^D$ represents the potential function of the s-th position of the sequence in the i-th CRF and $V_{j,s}^G$ represents the probability of the s-th position in the j-th HMM, that is, $V_{i,s}^D = \exp(\boldsymbol{\lambda}_i \cdot \boldsymbol{f}_s)$ and $V_{j,s}^G = \theta_{y_{s-1},y_s}\theta_{y_s,x_s}$, respectively. See the Appendix for the derivation of Equation (6) from Equation (2).

To estimate $\boldsymbol{\Gamma}^{(t+1)}$, namely procedure 4.2 in Figure 1, we employ the derivatives with respect to γ_i and γ_j shown in Equation (6), which are the parameters of the discriminative and generative models, respectively. Thus, we obtain the following derivatives with respect to γ_i:

$$\frac{\partial \mathcal{L}^{\text{HySOL}}(\boldsymbol{\Gamma}|\boldsymbol{\Theta})}{\partial \gamma_i} = \sum_n \log p_i^D(\boldsymbol{y}^n|\boldsymbol{x}^n) + \sum_n \log Z_i^D(\boldsymbol{x}^n) \\ - \sum_n E_{R(\mathcal{Y}|\boldsymbol{x}^n;\boldsymbol{\Lambda},\boldsymbol{\Theta},\boldsymbol{\Gamma})}\big[\sum_s \log V_{i,s}^D\big].$$

The first and second terms are constant during iterative procedure 4 in our optimization algorithm shown in Figure 1. Thus, we only need to calculate these values once at the beginning of procedure 4. Let $\boldsymbol{\alpha}_s(y)$ and $\boldsymbol{\beta}_s(y)$ represent the forward and backward state costs at position s with output y for corresponding input $\boldsymbol{x}$. Let $\mathcal{V}_s(y,y')$ represent the products of the total value of the transition cost between $s{-}1$ and s with labels y and y' in the corresponding input sequence, that is, $\mathcal{V}_s(y,y') = \prod_i [V_{i,s}^D(y,y')]^{\gamma_i} \prod_j [V_{j,s}^G(y,y')]^{\gamma_j}$. The third term, which indicates the expectation of potential functions, can be rewritten in the form of a forward-backward algorithm, that is,

$$E_{R(\mathcal{Y}|\boldsymbol{x};\boldsymbol{\Lambda},\boldsymbol{\Theta},\boldsymbol{\Gamma})}\big[\sum_s \log V_{i,s}^D\big] \\ = \frac{1}{Z_R(\boldsymbol{x})} \sum_s \sum_{y,y'} \boldsymbol{\alpha}_{s-1}(y) \mathcal{V}_s(y,y') \boldsymbol{\beta}_s(y') \log V_{i,s}^D(y,y'), \tag{7}$$

where $Z_R(\boldsymbol{x})$ represents the partition function of our hybrid model, that is, $Z_R(\boldsymbol{x}) {=} \mathcal{N}_R(\boldsymbol{x}) \prod_i [Z_i(\boldsymbol{x})]^{\gamma_i}$. Hence, the calculation of derivatives with respect to γ_i is tractable since we can incorporate the same forward-backward algorithm as that used in a standard CRF.

Then, the derivatives with respect to γ_j, which are the parameters of generative models, can be written as follows:

$$\frac{\partial \mathcal{L}^{\text{HySOL}}(\boldsymbol{\Gamma}|\boldsymbol{\Theta})}{\partial \gamma_j} \\ = \sum_n \log p_j^G(\boldsymbol{x}^n, \boldsymbol{y}^n) - \sum_n E_{R(\mathcal{Y}|\boldsymbol{x}^n;\boldsymbol{\Lambda},\boldsymbol{\Theta},\boldsymbol{\Gamma})}\big[\sum_s \log V_{j,s}^G\big].$$

Again, the second term, which indicates the expectation of transition probabilities and symbol emission probabilities, can be rewritten in the form of a forward-backward algorithm in the same manner as γ_i, where the only difference is that $V_{i,s}^D$ is substituted by $V_{j,s}^G$ in Equation (7).

To estimate $\boldsymbol{\Theta}^{(t+1)}$, which is procedure 4.1 in Figure 1, the same forward-backward algorithm as used in standard HMMs is available since the form of our Q-function shown in Equation (5) is the same as that of standard HMMs. The only difference is that our method uses marginal probabilities given by R instead of the $p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})$ of standard HMMs.

Therefore, only a forward-backward algorithm is required for the efficient calculation of our parameter estimation process. Note that even though our hybrid model supports the use of a combination of several generative and discriminative models, we only need to calculate the forward-backward algorithm once for each sample during optimization procedures 4.1 and 4.2. This means that the required number of executions of the forward-backward algorithm for our parameter estimation is independent of the number of models used in the hybrid model.

In addition, after training, we can easily merge all the parameter values in a single parameter vector. This means that we can simply employ the Viterbi-algorithm for evaluating unseen samples, as well as that of standard CRFs, without any additional cost.

4 Experiments

We examined our hybrid model (HySOL) by applying it to two sequence labeling tasks, named entity recognition (NER) and syntactic chunking (Chunking). We used the same Chunking and 'English' NER data as those used for the shared tasks of CoNLL-2000 (Tjong Kim Sang and Buchholz, 2000) and CoNLL-2003 (Tjong Kim Sang and Meulder, 2003), respectively.

For the baseline method, we performed a conditional random field (CRF), which is exactly the same training procedure described in (Sha and Pereira, 2003) with L-BFGS. Moreover, LOP-CRF (Smith et al., 2005) is also compared with our hybrid model, since the formalism of our hybrid model can be seen as an extension of LOP-CRFs as described in Section 3. For CRF, we used the Gaussian prior as the second term on the RHS in Equation (1), where δ^2 represents the hyper-parameter in the Gaussian prior. In contrast, for LOP-CRF and HySOL, we used the Dirichlet priors as the second term on the

$\boldsymbol{\lambda}_1$	f(word$_s$), f(lword$_s$), f(pos$_s$), f(wtype$_s$), f(pos$_{s-1}$, pos$_s$), f(wtype$_{s-1}$, wtype$_s$), f(pos$_s$, pos$_{s+1}$), f(wtype$_s$, wtype$_{s+1}$), f(pref1$_s$), f(pref2$_s$), f(pref3$_s$), f(pref4$_s$), f(suf1$_s$), f(suf2$_s$), f(suf3$_s$), f(suf4$_s$)
$\boldsymbol{\lambda}_2$	f(word$_s$), f(lword$_s$), f(pos$_s$), f(wtype$_s$), f(word$_{s-1}$), f(lword$_{s-1}$), f(pos$_{s-1}$), f(wtype$_{s-1}$), f(word$_{s-2}$), f(lword$_{s-2}$), f(pos$_{s-2}$), f(wtype$_{s-2}$), f(pos$_{s-2}$, pos$_{s-1}$), f(wtype$_{s-2}$, wtype$_{s-1}$)
$\boldsymbol{\lambda}_3$	f(word$_s$), f(lword$_s$), f(pos$_s$), f(wtype$_s$), f(word$_{s+1}$), f(lword$_{s+1}$), f(pos$_{s+1}$), f(wtype$_{s+1}$), f(word$_{s+2}$), f(lword$_{s+2}$), f(pos$_{s+2}$), f(wtype$_{s+2}$), f(pos$_{s+1}$, pos$_{s+2}$), f(wtype$_{s+1}$, wtype$_{s+2}$)
$\boldsymbol{\lambda}_4$	all of the above

lword : lowercase of word,   wtype : 'word type'
pref1-4: 1-4 character prefix of word
suf1-4 : 1-4 character suffix of word

Table 1: Features used in NER experiments

RHS in Equations (3), and (4), where ξ and η are the hyper-parameters in each Dirichlet prior.

4.1 Named Entity Recognition Experiments

The English NER data consists of 203,621, 51,362 and 46,435 words from 14,987, 3,466 and 3,684 sentences in training, development and test data, respectively, with four named entity tags, PERSON, LOCATION, ORGANIZATION and MISC, plus the 'O' tag. The unlabeled data consists of 17,003,926 words from 1,029,122 sentences. These data sets are exactly the same as those provided for the shared task of CoNLL-2003.

We slightly extended the feature set of the supplied data by adding feature types such as 'word type', and word prefix and suffix. Examples of 'word type' include whether the word is capitalized, contains digit or contains punctuation, which basically follows the baseline features of (Sutton et al., 2006) without regular expressions. Note that, unlike several previous studies, we did not employ additional information from external resources such as gazetteers. All our features can be automatically extracted from the supplied data.

For LOP-CRF and HySOL, we used four base discriminative models trained by CRFs with different feature sets. Table 1 shows the feature sets we used for training these models. The design of these feature sets was derived from a suggestion in (Smith et al., 2005), which exhibited the best performance in the several feature division. Note that the CRF for the comparison method was trained by using all feature types, namely the same as $\boldsymbol{\lambda}_4$.

$\boldsymbol{\lambda}_1$	f(word$_s$), (pos$_s$), f(word$_{s-1}$, word$_s$), f(pos$_{s-1}$, pos$_s$), f(word$_s$, word$_{s+1}$), f(pos$_s$, pos$_{s+1}$)
$\boldsymbol{\lambda}_2$	f(word$_s$), (pos$_s$), f(word$_{s-1}$), f(pos$_{s-1}$), f(word$_{s-2}$), f(pos$_{s-2}$), f(word$_{s-2}$, word$_{s-1}$), f(pos$_{s-2}$, pos$_{s-1}$)
$\boldsymbol{\lambda}_3$	f(word$_s$), (pos$_s$), f(word$_{s+1}$), f(pos$_{s+1}$), f(word$_{s+2}$), f(pos$_{s+2}$), f(word$_{s+1}$, word$_{s+2}$), f(pos$_{s+1}$, pos$_{s+2}$)
$\boldsymbol{\lambda}_4$	all of the above

Table 2: Features used in Chunking experiments

As we explained in Section 3.3, for training HySOL, the parameters of four discriminative models, $\boldsymbol{\Lambda}$, were trained from 4/5 of the labeled training data, and $\boldsymbol{\Gamma}$ were trained from remaining 1/5. For the features of the generative models, we used all of the feature types shown in Figure 1. Note that one feature type corresponds to one HMM. Thus, each HMM maintains to consist of a non-overlapping feature set since each feature type only generates one symbol per state.

4.2 Syntactic Chunking Experiments

CoNLL-2000 Chunking data was obtained from the Wall Street Journal (WSJ) corpus: sections 15-18 as training data (8,936 sentences and 211,727 words), and section 20 as test data (2,012 sentences and 47,377 words), with 11 different chunk-tags, such as NP and VP plus the 'O' tag, which represents the region outside any target chunk.

For LOP-CRF and HySOL, we also used four base discriminative models trained by CRFs with different feature sets. Table 2 shows the feature set we used in the Chunking experiments. We used the feature set of the supplied data without any extension of additional feature types.

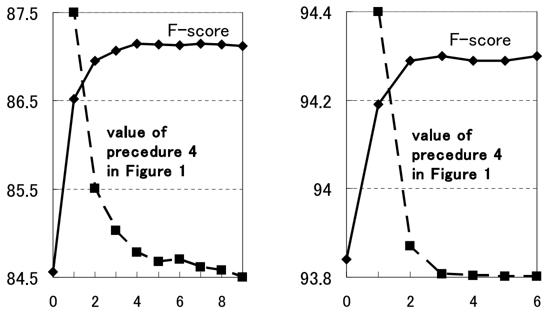
To train HySOL, we used the same unlabeled data as used for our NER experiments (17,003,926 words from the Reuters corpus). Moreover, the division of the labeled training data and the feature set of the generative models were derived in the same manner as our NER experiments (see Section 4.1). That is, we divided the labeled training data into 4/5 for estimating $\boldsymbol{\Lambda}$ and 1/5 for estimating $\boldsymbol{\Gamma}$; one feature type shown in Table 2 is assigned in one generative model.

methods	(hyper-params)	$F_{\beta=1}$	(gain)	Sent	(gain)
CRF	(δ^2=100.0)	84.70	-	78.30	-
(4/5 labeled data, δ^2=100.0)		83.74	(-0.96)	77.06	(-1.24)
LOP-CRF	(ξ'=0.1)	84.90	(+0.20)	79.02	(+0.72)
HySOL	(ξ'=0.1,η'=0.0001)	**87.20**	(+2.50)	**81.19**	(+2.89)
	(w/o prior)	86.86	(+2.16)	80.75	(+2.45)
w/o $p_j^G \ \forall j$	(ξ'=1.0)	84.56	(-0.14)	78.23	(-0.07)

Table 3: NER performance (CoNLL-2003)

methods	(hyper-params)	$F_{\beta=1}$	(gain)	Sent	(gain)
CRF	(δ^2=10.0)	93.87	-	59.84	-
(4/5 labeled data, δ^2=10.0)		93.70	(-0.17)	58.85	(-0.99)
LOP-CRF	(ξ'=0.1)	93.91	(+0.04)	60.34	(+0.50)
HySOL	(ξ'=1.0,η'=0.0001)	**94.30**	(+0.43)	**61.73**	(+1.89)
	(w/o prior)	94.17	(+0.30)	61.23	(+1.39)
w/o $p_j^G \ \forall j$	(ξ'=1.0)	93.84	(-0.03)	59.74	(-0.10)

Table 4: Chunking performance (CoNLL-2000)



(a) NER (b) Chunking

Figure 2: Changes in the performance and the convergence condition value (procedure 4 in Figure 1) of HySOL.

5 Results and Discussion

We evaluated the performance in terms of the $F_{\beta=1}$ score, which is the evaluation measure used in CoNLL-2000 and 2003, and sentence accuracy, since all the methods in our experiments optimize sequence loss. Tables 3 and 4 show the results of the NER and Chunking experiments, respectively. The $F_{\beta=1}$ and 'Sent' columns show the performance evaluated using the $F_{\beta=1}$ score and sentence accuracy, respectively. δ^2, ξ and η, which are the hyper-parameters in Gaussian or Dirichlet priors, are selected from a certain value set by using a development set[1], that is, $\delta^2 \in \{0.01, 0.1, 1, 10, 100, 1000\}$, $\xi - 1 = \xi' \in \{0.01, 0.1, 1, 10\}$ and $\eta - 1 = \eta' \in \{0.00001, 0.0001, 0.001, 0.01\}$. The second rows of CRF in Tables 3 and 4 represent the performance of base discriminative models used in HySOL with all the features, which are trained with 4/5 of the labeled training data. The third rows of HySOL show the performance obtained without using generative models (unlabeled data). The model itself is essentially the same as LOP-CRFs. However the performance in the third HySOL rows was consistently lower than that of LOP-CRF since the discriminative models in HySOL are trained with 4/5 labeled data.

As shown in Tables 3 and 4, HySOL significantly improved the performance of supervised setting, CRF and LOP-CRF, as regards both NER and Chunking experiments.

5.1 Impact of Incorporating Unlabeled Data

The contributions provided by incorporating unlabeled data in our hybrid model can be seen by comparison with the performance of the first and third rows in HySOL, namely a 2.64 point F-score and a 2.96 point sentence accuracy gain in the NER experiments and a 0.46 point F-score and a 1.99 point sentence accuracy gain in the Chunking experiments.

We believe there are two key ideas that enable the unlabeled data in our approach to exhibit this improvement compared with the the state-of-the-art performance provided by discriminative models in supervised settings. First, unlabeled data is only used for optimizing Equation (4) to obtain a similar effect to 'soft-clustering', which can be calculated without information about the correct output. Second, by using a combination of generative models, we can enhance the flexibility of the feature design for unlabeled data. For example, we can handle arbitrary overlapping features, similar to those used in discriminative models, for unlabeled data by assigning one feature type for one generative model as in our experiments.

5.2 Impact of Iterative Parameter Estimation

Figure 2 shows the changes in the performance and the convergence condition value of HySOL during parameter estimation iteration in our NER and Chunking experiments, respectively. As shown in the figure, HySOL was able to reach the conver-

[1] Chunking (CoNLL-2000) data has no common development set. Thus, our preliminary examination employed by using 4/5 labeled training data with the remaining 1/5 as development data to determine the hyper-parameter values.

gence condition in a small number of iterations in our experiments. Moreover, the change in the performance remains quite stable during the iteration. However, theoretically, our optimization procedure is not guaranteed to converge in the $\mathbf{\Gamma}$ and $\mathbf{\Theta}$ space, since the optimization of $\mathbf{\Theta}$ has local maxima. Even if we were unable to meet the convergence condition, we were easily able to obtain model parameters by performing a sufficient fixed number of iterations, and then select the parameters when Equation (4) obtained the maximum objective value.

5.3 Comparison with SS-CRF-MER

When we consider semi-supervised SOL methods, SS-CRF-MER (Jiao et al., 2006) is the most competitive with HySOL, since both methods are defined based on CRFs. We planned to compare the performance with that of SS-CRF-MER in our NER and Chunking experiments. Unfortunately, we failed to implement SS-CRF-MER since it requires the use of a slightly complicated algorithm, called the ‘nested’ forward-backward algorithm.

Although, we cannot compare the performance, our hybrid approach has several good characteristics compared with SS-CRF-MER. First, it requires a higher order algorithm, namely a ‘nested’ forward-backward algorithm, for the parameter estimation of unlabeled data whose time complexity is $O(L^3S^2)$ for each unlabeled data, where L and S represent the output label size and unlabeled sample length, respectively. Thus, our hybrid approach is more scalable for the size of unlabeled data, since HySOL only needs a standard forward-backward algorithm whose time complexity is $O(L^2S)$. In fact, we still have a question as to whether SS-CRF-MER is really scalable in practical time for such a large amount of unlabeled data as used in our experiments, which is about 680 times larger than that of (Jiao et al., 2006). Scalability for unlabeled data will become really important in the future, as it will be natural to use millions or billions of unlabeled data for further improvement. Second, SS-CRF-MER has a sensitive hyper-parameter in the objective function, which controls the influence of the unlabeled data. In contrast, our objective function only has a hyper-parameter of prior distribution, which is widely used for standard MAP estimation. Moreover, the experimental results shown in Tables 3 and 4 indicate that HySOL is rather robust with respect to the hyper-parameter since we can obtain fairly good performance without a prior distribution.

	$\mathrm{F}_{\beta=1}$	additional resources
ASO-semi (Ando and Zhang, 2005)	89.31	unlabeled data (27M words)
(Florian et al., 2003)	88.76	their own large gazetteers, 2M-word labeled data
(Chieu and Ng, 2003)	88.31	their own large gazetteers, very elaborated features
HySOL	**88.14**	unlabeled data (17M words) supplied gazetters
HySOL	**87.20**	unlabeled data (17M words)

Table 5: Previous top systems in NER (CoNLL-2003) experiments

	$\mathrm{F}_{\beta=1}$	additional resources
ASO-semi (Ando and Zhang, 2005)	94.39	unlabeled data (15M words: WSJ)
HySOL	**94.30**	unlabeled data (17M words: Reuters)
(Zhang et al., 2002)	94.17	full parser output
(Kudo and Matsumoto, 2001)	93.91	–

Table 6: Previous top systems in Chunking (CoNLL-2000) experiments

5.4 Comparison with Previous Top Systems

With respect to the performance of NER and Chunking tasks, the current best performance is reported in (Ando and Zhang, 2005), which we refer to as ‘ASO-semi’, as shown in Figures 5 and 6. ASO-semi also incorporates unlabeled data solely for the additional information in the same way as our method. Unfortunately, our results could not reach their level of performance, although the size and source of the unlabeled data are not the same for certain reasons. First, (Ando and Zhang, 2005) does not describe the unlabeled data used in their NER experiments in detail, and second, we are not licensed to use the TREC corpus including WSJ unlabeled data that they used for their Chunking experiments (training and test data for Chunking is derived from WSJ). Therefore, we simply used the supplied unlabeled data of the CoNLL-2003 shared task for both NER and Chunking. If we consider the advantage of our approach, our hybrid model incorporating generative models seems rather intuitive, since it is sometimes difficult to find out a design of effective auxiliary problems for the target problem.

Interestingly, the additional information obtained

	$F_{\beta=1}$	(gain)
HySOL (ξ'=0.1,η'=0.0001)	87.20	-
+ w/ F-score opt. (Suzuki et al., 2006)	88.02	(+0.82)
+ unlabeled data (17M → 27M words)	88.41	(+0.39)
+ supplied gazetters	88.90	(+0.49)
+ add dev. set for estimating $\mathbf{\Gamma}$	89.27	(+0.37)

Table 7: The HySOL performance with the F-score optimization technique and some additional resources in NER (CoNLL-2003) experiments

	$F_{\beta=1}$	(gain)
HySOL (ξ'=0.1,η'=0.0001)	94.30	-
+ w/ F-score opt. (Suzuki et al., 2006)	94.36	(+0.06)

Table 8: The HySOL performance with the F-score optimization technique on Chunking (CoNLL-2000) experiments

from unlabeled data appear different from each other. ASO-semi uses unlabeled data for constructing auxiliary problems to find the 'shared structures' of auxiliary problems that are expected to improve the performance of the main problem. Moreover, it is possible to combine both methods, for example, by incorporating the features obtained with their method in our base discriminative models, and then construct a hybrid model using our method. Therefore, there may be a possibility of further improving the performance by this simple combination.

In NER, most of the top systems other than ASO-semi boost performance by employing external hand-crafted resources such as large gazetteers. This is why their results are superior to those obtained with HySOL. In fact, if we simply add the gazetteers included in CoNLL-2003 supplied data as features, HySOL achieves 88.14.

5.5 Applying F-score Optimization Technique

In addition, we can simply apply the F-score optimization technique for the sequence labeling tasks proposed in (Suzuki et al., 2006) to boost the HySOL performance since the base discriminative models $p^D(\boldsymbol{y}|\boldsymbol{x})$ and discriminative combination, namely Equation (3), in our hybrid model basically uses the same optimization procedure as CRFs. Tables 7 and 8 show the F-score gain when we apply the F-score optimization technique. As shown in the Tables, the F-score optimization technique can easily improve the (F-score) performance without any additional resources or feature engineering.

In NER, we also examined HySOL with additional resources to observe the performance gain. The third row represents the performance when we add approximately 10M words of unlabeled data (total 27M words)[2] that are derived from 1996/11/15-30 articles in Reuters corpus. Then, the fourth and fifth rows represent the performance when we add the supplied gazetters in the CoNLL-2003 data as features, and adding development data as training data of $\mathbf{\Gamma}$. In this case, HySOL achieved a comparable performance to that of the current best system, ASO-semi, in both NER and Chunking experiments even though the NER experiment is not a fair comparison since we added additional resources (gazetters and dev. set) that ASO-semi does not use in training.

6 Conclusion and Future Work

We proposed a framework for semi-supervised SOL based on a hybrid generative and discriminative approach. Experimental results showed that incorporating unlabeled data in a generative manner has the power to further improve on the state-of-the-art performance provided by supervised SOL methods such as CRFs, with the help of our hybrid approach, which discriminatively combines with discriminative models. In future we intend to investigate more appropriate model and feature design for unlabeled data, which may further improve the performance achieved in our experiments.

Appendix

Let $V^D_{i,s} = \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}_s)$ and $V^G_{j,s} = \theta_{y_{s-1},y_s}\theta_{y_s,x_s}$. Equation (6) can be obtained by the following rearrangement of Equation (2) :

$$
\begin{aligned}
& R(\boldsymbol{y}|\boldsymbol{x};\mathbf{\Lambda},\mathbf{\Theta},\mathbf{\Gamma}) \\
&= \frac{\prod_i p^D_i(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}_i)^{\gamma_i} \prod_j p^G_j(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta}_j)^{\gamma_j}}{\sum_{\boldsymbol{y}} \prod_i p^D_i(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}_i)^{\gamma_i} \prod_j p^G_j(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta}_j)^{\gamma_j}} \\
&= \frac{1}{\mathcal{N}_R(\boldsymbol{x})} \prod_i [\frac{\prod_s V^D_{i,s}}{Z_i(\boldsymbol{x})}]^{\gamma_i} \prod_j [\prod_s V^G_{j,s}]^{\gamma_j} \\
&= \frac{1}{\mathcal{N}_R(\boldsymbol{x}) \prod_i [Z_i(\boldsymbol{x})]^{\gamma_i}} \prod_i [\prod_s V^D_{i,s}]^{\gamma_i} \prod_j [\prod_s V^G_{j,s}]^{\gamma_j} \\
&= \frac{1}{\mathcal{N}_R(\boldsymbol{x}) \prod_i [Z_i(\boldsymbol{x})]^{\gamma_i}} \prod_s \prod_i [V^D_{i,s}]^{\gamma_i} \prod_j [V^G_{j,s}]^{\gamma_j}.
\end{aligned}
$$

[2] In order to keep the consistency of POS tags, we re-attached POS tags of the supplied data set and new 10M words of unlabeled data using a POS tagger trained from WSJ corpus.

References

Y. Altun, D. McAllester, and M. Belkin. 2005. Maximum Margin Semi-Supervised Learning for Structured Variables. In *Proc. of NIPS*2005*.

R. Ando and T. Zhang. 2005. A High-Performance Semi-Supervised Learning Method for Text Chunking. In *Proc. of ACL-2005*, pages 1–9.

U. Brefeld and T. Scheffer. 2006. Semi-Supervised Learning for Structured Output Variables. In *Proc. of ICML-2006*.

H. L. Chieu and Hwee T. Ng. 2003. Named Entity Recognition with a Maximum Entropy Approach. In *Proc. of CoNLL-2003*, pages 160–163.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named Entity Recognition through Classifier Combination. In *Proc. of CoNLL-2003*, pages 168–171.

A. Fujino, N. Ueda, and K. Saito. 2005. A Hybrid Generative/Discriminative Approach to Semi-Supervised Classifier Design. In *Proc. of AAAI-05*, pages 764–769.

Y. Grandvalet and Y. Bengio. 2004. Semi-Supervised Learning by Entropy Minimization. In *Proc. of NIPS*2004*, pages 529–536.

F. Jiao, S. Wang, C.-H. Lee, R. Greiner, and D. Schuurmans. 2006. Semi-Supervised Conditional Random Fields for Improved Sequence Segmentation and Labeling. In *Proc. of COLING/ACL-2006*, pages 209–216.

T. Kudo and Y. Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of NAACL 2001*, pages 192–199.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML-2001*, pages 282–289.

W. Li and A. McCallum. 2005. Semi-Supervised Sequence Modeling with Syntactic Topic Models. In *Proc. of AAAI-2005*, pages 813–818.

D. C. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Programming, Ser. B*, 45(3):503–528.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39:103–134.

R. Raina, Y. Shen, A. Y. Ng, and A. McCallum. 2003. Classification with Hybrid Generative/Discriminative Models. In *Proc. of NIPS*2003*.

F. Sha and F. Pereira. 2003. Shallow Parsing with Conditional Random Fields. In *Proc. of HLT/NAACL-2003*, pages 213–220.

A. Smith, T. Cohn, and M. Osborne. 2005. Logarithmic Opinion Pools for Conditional Random Fields. In *Proc. of ACL-2005*, pages 10–17.

C. Sutton, M. Sindelar, and A. McCallum. 2006. Reducing Weight Undertraining in Structured Discriminative Learning. In *Proc. of HTL-NAACL 2006*, pages 89–95.

J. Suzuki, E. McDermott, and H. Isozoki. 2006. Training Conditional Random Fields with Multivariate Evaluation Measure. In *Proc. of COLING/ACL-2006*, pages 217–224.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proc. of CoNLL-2000 and LLL-2000*, pages 127–132.

E. T. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proc. of CoNLL-2003*, pages 142–147.

T. Zhang, F. Damerau, and D. Johnson. 2002. Text Chunking based on a Generalization of Winnow. *Machine Learning Research*, 2:615–637.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-Supervised Learning using Gaussian Fields and Harmonic Functions. In *Proc.of ICML-2003*, pages 912–919.

Finding Good Sequential Model Structures using Output Transformations

Edward Loper
Department of Computer & Information Science
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19104
edloper@cis.upenn.edu

Abstract

In Sequential Viterbi Models, such as HMMs, MEMMs, and Linear Chain CRFs, the type of patterns over output sequences that can be learned by the model depend directly on the model's structure: any pattern that spans more output tags than are covered by the models' order will be very difficult to learn. However, increasing a model's order can lead to an increase in the number of model parameters, making the model more susceptible to sparse data problems.

This paper shows how the notion of *output transformation* can be used to explore a variety of alternative model structures. Using output transformations, we can selectively increase the amount of contextual information available for some conditions, but not for others, thus allowing us to capture longer-distance consistencies while avoiding unnecessary increases to the model's parameter space. The appropriate output transformation for a given task can be selected by applying a hill-climbing approach to held-out data. On the NP Chunking task, our hill-climbing system finds a model structure that outperforms both first-order and second-order models with the same input feature set.

1 Sequence Prediction

A *sequence prediction task* is a task whose input is a sequence and whose output is a corresponding sequence. Examples of sequence prediction tasks include part-of-speech tagging, where a sequence of words is mapped to a sequence of part-of-speech tags; and IOB noun phrase chunking, where a sequence of words is mapped to a sequence of labels, I, O, and B, indicating whether each word is inside a chunk, outside a chunk, or at the boundary between two chunks, respectively.

In sequence prediction tasks, we are interested in finding the most likely output sequence for a given input. In order to be considered likely, an output value must be consistent with the input value, but it must also be internally consistent. For example, in part-of-speech tagging, the sequence "preposition-verb" is highly unlikely; so we should reject an output value that contains that sequence, even if the individual tags are good candidates for describing their respective words.

2 Sequential Viterbi Models

This intuition is captured in many sequence learning models, including Hidden Markov Models (HMMs), Maximum Entropy Markov Models (MEMMs), and Linear Chain Conditional Random Fields (LC-CRFs), by including terms corresponding to pieces of output structure in their scoring functions. (Sha and Pereira, 2003; Sutton and McCallum, 2006; McCallum et al., 2000; Alpaydin, 2004)

Each of these *Sequential Viterbi Models* defines a set of scoring functions that evaluate fixed-size pieces of the output sequence based on fixed-size pieces of the input sequence.[1] The overall score for

[1] For HMMs and MEMMs, the local scores are negative log probabilities. For LC-CRFs, the local scores do not have any direct probabilistic interpretation.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 801–809, Prague, June 2007. ©2007 Association for Computational Linguistics

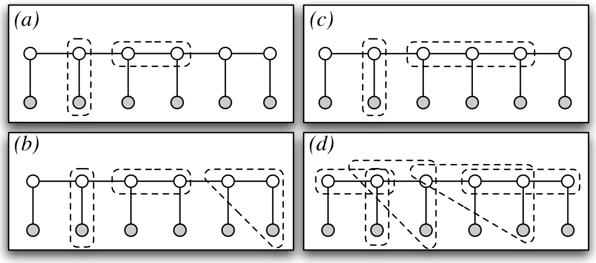


Figure 1: **Common Model Structures**. *(a)* Simple first order. *(b)* Extended first order. *(c)* Simple second order. *(d)* Extended second order.

an output value is then computed by summing the scores for all its fixed-size pieces. Sequence prediction models can differ from one another along two dimensions:

1. **Model Structure:** The set of output pieces and input pieces for which local scoring functions are defined.

2. **Model Type:** The set of parametrized equations used to define those local scoring functions, and the procedures used to determine their parameters.

In this paper, we focus on model structure. In particular, we are interested in finding a suitable model structure for a given task and training corpus.

2.1 Common Model Structures

The model structure used by classical HMMs is the "simple first order" structure. This model structure defines two local scoring functions. The first scoring function evaluates an output value in the context of the corresponding input value; and the second scoring function evaluates adjacent pairs of output values. Simple LC-CRFs often extend this structure by adding a third local scoring function, which evaluates adjacent pairs of output values in the context of the input value corresponding to one of those outputs. These model structures are illustrated in Figure 1.

Because these first order structures include scoring functions for adjacent pairs of output items, they can identify and reject output values that contain improbable subsequences of length two. For example, in part-of-speech tagging, the sequence "preposition-verb" is highly unlikely; and such models will easily learn to reject outputs containing that sequence. However, it is much more difficult for first order models to identify improbable subsequences of length three or more. For example, in English texts, the sequence "verb-noun-verb" is much less likely than one would predict based just on the subsequences "verb-noun" and "noun-verb." But first order models are incapable of learning that fact.

Thus, in order to improve performance, it is often necessary to include scoring functions that span over larger sequences. In the "simple second order" model structure, the local scoring function for adjacent pairs of output values is replaced with a scoring function for each triple of consecutive output values. In extended versions of this structure typically used by LC-CRFs, scoring functions are also added that combine output value triples with an input value. These model structures are illustrated in Figure 1. Similarly, third order and and fourth order models can be used to further increase the span over which scoring functions are defined.

Moving to higher order model structures increases the distance over which the model can check consistency. However, it also increases the number of parameters the model must learn, making the model more susceptible to sparse data problems. Thus, the usefulness of a model structure for a given task will depend on the types of constraints that are important for the task itself, and on the size and diversity of the training corpus.

3 Searching for Good Model Structures

We can therefore use simple search methods to look for a suitable model structure for a given task and training corpus. In particular, we have performed several experiments using hill-climbing methods to search for an appropriate model structure for a given task. In order to apply hill-climbing methods, we need to define:

1. The search space. I.e., concrete representations for the set of model structures we will consider.

2. A set of operations for moving through that search space.

3. An evaluation metric.

In Section 4, we will define the search space using transformations on output values. This will allow us to consider a wide variety of model structures without needing to make any direct modifications to the underlying sequence modelling systems. Output value transformations will be concretely represented using Finite State Transducers (FSTs). In Section 5, we will define the set of operations for moving through the search space as modification operations on FSTs. For the evaluation metric, we simply train and test the model, using a given model structure, on held-out data.

4 Representing Model Structure with Reversible Output Transformations

The common model structures described in Section 2.1 differ from one another in that they examine varying sizes of "windows" on the output structure. Rather than varying the size of the window, we can achieve the same effect by fixing the window size, but transforming the output values. For example, consider the effects of transforming the output values by replacing individual output tags with pairs of adjacent output tags:

$$y_1, y_2, \ldots, y_t \rightarrow \langle \mathrm{START}, y_1\rangle, \langle y_1, y_2\rangle, \langle y_2, y_3\rangle, \ldots, \langle y_{t-1}, y_t\rangle$$

E.g.:

```
   I   O   O   I   I   B   I  →
  OI  IO  OO  OI  II  IB  BI
```

Training a first order model based on these transformed values is equivalent to training a second order model based on the original values, since in each case the local scoring functions will be based on pairs of adjacent output tags. Similarly, transforming the output values by replacing individual output tags with triples of adjacent output tags is equivalent to training a third order model based on the original output values.

Of course, when we apply a model trained on this type of transformed output to new inputs, it will generate transformed output values. Thus, the transformation must be reversible, so that we can map the output of the model back to an un-transformed output value.

This transformational approach has the advantage that we can explore different model structures using off-the-shelf learners, without modifying them. In particular, we can apply the transformation corresponding to a given model structure to the training corpus, and then train the off-the-shelf learner based on that transformed corpus. To predict the value for a new input, we simply apply the learned model to generate a corresponding transformed output value, and then use the inverse transformation to map that value back to an un-transformed value.

Output encoding transformations can be used to represent a large class of model structures, including commonly used structures (first order, second order, etc) as well as a number of "hybrid" structures that use different window sizes depending on the content of the output tags.

Output encoding transformations can also be used to represent a wide variety of other model structures. For example, there has been some debate about the relative merits of different output encodings for the chunking task (Tjong Kim Sang and Veenstra, 1999; Tjong Kim Sang, 2000; Shen and Sarkar, 2005). These encodings differ in whether they define special tags for the beginning of chunks, for the ends of chunks, and for boundaries between chunks. The output transformation procedure described here is capable of capturing all of the output encodings used for chunking. Thus, this transformational method provides a unified framework for considering both the type of information that should be encoded by individual tags (i.e., the encoding) and the distance over which that information should be evaluated (i.e., the order of the model). Under this framework, we can use simple search procedures to find an appropriate transformation for a given task.

4.1 Representing Transformations as FSTs

Finite State Transducers (FSTs) provide a natural formalism for representing output transformations. FSTs are powerful enough to capture different orders of model structure, including hybrid orders; and to capture different output encodings, such as the ones considered in (Shen and Sarkar, 2005). FSTs are efficient, so they add very little overhead. Finally, there exist standard algorithms for inverting

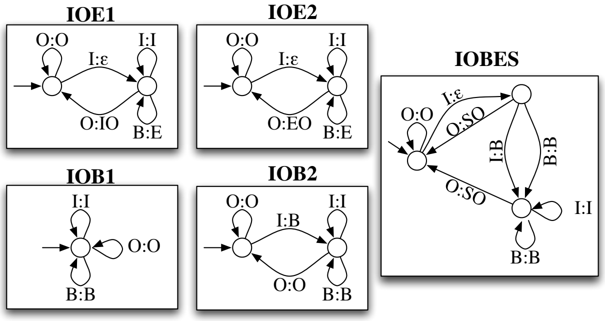


Figure 2: **FSTs for Five Common Chunk Encodings**. Each transducer takes an `IOB1`-encoded string for a given output value, and generates the corresponding string for the same output value, using a new encoding. Note that the `IOB1` FST is an identity transducer; and note that the transducers that make use of the `E` tag must use ϵ-output edges to delay the decision of which tag should be used until enough information is available.

and determinizing FSTs. [2]

4.1.1 Necessary Properties for Output-Transformation FSTs

In order for an FST to be used to transform output values, it must have the following three properties:

1. The FST's inverse should be deterministic.[3] Otherwise, we will be unable to convert the model's (transformed) output into an untransformed output value.

2. The FST should recognize exactly the set of valid output values. If it does not recognize some valid output value, then it won't be able to transform that value. If it recognizes some invalid output value, then there exists an transformed output value that would map back to an invalid output value.

3. The FST should not modify the length of the output sequence. Otherwise, it will not be possible to align the output values with input values when running the model.

In addition, it seems desirable for the FST to have the following two properties:

4. The FST should be deterministic. Otherwise, a single training example's output could be encoded in multiple ways, which would make training the individual base decision classifiers difficult.

5. The FST should generate every output string. Otherwise, there would be some possible system output that we are unable to map back to an un-transformed output.

Unfortunately, these two properties, when taken together with the first three, are problematic. To see why, assume an FST with an output alphabet of size k. Property (5) requires that all possible output strings be generated, and property (1) requires that no string is generated for two input strings, so the number of strings generated for an input of length n must be exactly k^n. But the number of possible chunkings for an input of length n is $3^n - 3^{n-1} - 3^{n-2}$; and there is no integer k such that $k^n = 3^n - 3^{n-1} - 3^{n-2}$.[4]

We must therefore relax at least one of these two properties. Relaxing the property 4 (deterministic FSTs) will make training harder; and relaxing the property 5 (complete FSTs) will make testing harder. In the experiments presented here, we chose to relax the second property.

4.1.2 Inverting the Transformation

Recall that the motivation behind property 5 is that we need a way to map *any* output generated by the machine learning system back to an untransformed output value.

As an alternative to requiring that the FST generate every output string, we can define an extended inversion function, that includes the inverted FST, but also generates output values for transformed values that are not generated by the FST. In particular,

[2] Note that we are not attempting to learn a transducer that generates the output values from input values, as is done in e.g. (Oncina et al., 1993) and (Stolcke and Omohundro, 1993). Rather, we we are interested in finding a transducer from one output encoding to another output encoding that will be more amenable to learning by the underlying Sequential Viterbi Model.

[3] Or at least determinizable.

[4] To see why the number of possible chunkings is $3^n - 3^{n-1} - 3^{n-2}$, consider the IOB1 encoding: it generates all chunkings, and is valid for any of the 3^n strings except those that start with `B` (of which there are 3^{n-1}) and those that include the sequence `OB` (of which there are 3^{n-2}).

in cases where the transformed value is not generated by the FST, we can assume that one or more of the transformed tags was chosen incorrectly; and make the minimal set of changes to those tags that results in a string that *is* generated by the FST. Thus, we can compute the optimal un-transformed output value corresponding to each transformed output using the following procedure:

1. Invert the original FST. I.e., replace each arc $\langle S \rightarrow Q[\alpha : \beta] \rangle$ with an arc $\langle S \rightarrow Q[\beta : \alpha] \rangle$.
2. Normalize the FST such that each arc has exactly one input symbol.
3. Convert the FST to a weighted FST by assigning a weight of zero to all arcs. This weighted FST uses non-negative real-valued weights, and the weight of a path is the sum of the weights of all edges in that path.
4. For each arc $\langle S \rightarrow Q[x : \alpha] \rangle$, and each $y \neq x$, add a new arc $\langle S \rightarrow Q[y : \alpha] \rangle$ with a weight one.
5. Determinize the resulting FST, using a variant of the algorithm presented in (Mohri, 1997). This determinization algorithm will prune paths that have non-optimal weights. In cases where determinization algorithm has not completed by the time it creates 10,000 states, the candidate FST is assumed to be non-determinizable, and the original FST is rejected as a candidate.

The resulting FST will accept all sequences of transformed tags, and will generate for each transformed tag the un-transformed output value that is generated with the fewest number of "repairs" made to the transformed tags.

5 FST Modification Operations

In order to search the space of output-transforming FSTs, we must define a set of modification operations, that generate a new FST from a previous FST. In order to support a hill-climbing search strategy, these modification operations should make small incremental changes to the FSTs. The selection of appropriate modification operations is important, since it will significantly impact the efficiency of the search process. In this section, I describe the set of FST modification operations that are used for the experiments described in this paper. These operations were chosen based our intuitions about what modifications would support efficient hill-climbing search. In future experiments, we plan to examine alternative modification operations.

5.1 New Output Tag

The *new output tag* operation replaces an arc $\langle S \rightarrow Q[\alpha : \beta x \gamma] \rangle$ with an arc $\langle S \rightarrow Q[\alpha : \beta y \gamma] \rangle$, where y is a new output tag that is not used anywhere else in the transducer. When a single output tag appears on multiple arcs, this operation effectively splits that tag in two. For example, when applied to the identity transducer for the IOB1 encoding shown in Figure 2, this operation can be used to distinguish O tags that follow other O tags from O tags that follow I or B tags – effectively increasing the order of the model structure for just O tags.

5.2 Specialize Output Tag[5]

The *specialize output tag* operation is similar to the *new output tag* operation, but rather than replacing the output tag with a new tag, we "subdivide" the tag. When the model is trained, features will be included for both the subdivided tag and the original (undivided) tag.

5.3 Loop Unrolling

The *loop unrolling* operation acts on a single self-loop arc e at a state S, and makes the following changes to the FST:

1. Create a new state S'.
2. For each outgoing arc $e_1 = \langle S \rightarrow Q[\alpha : \beta] \rangle \neq e$, add add an arc $e_2 = \langle S' \rightarrow Q[\alpha : \beta] \rangle$. Note that if e_1 was a self-loop arc (i.e., $S = Q$), then e_2 will point from S' to S.
3. Change the destination of loop arc e from S to S'.

By itself, the *loop unrolling* operation just modifies the structure of the FST, but does not change

[5] This operation requires the use of a model where features are defined over (input,output) pairs, such as MEMMs or LC-CRFs.

the actual transduction performed by the FST. It is therefore always immediately followed by applying the *new output tag* operation or the *specialize output tag* operation to the loop arc e.

5.4 Copy Tag Forward

The *copy tag forward* operation splits an existing state in two, directing all incoming edges that generate a designated output tag to one copy, and all remaining incoming edges to the other copy. The outgoing edges of these two states are then distinguished from one another, using either the *specialize output tag* operation (if available) or the *new output tag* operation.

This modification operation creates separate edges for different output histories, effectively increasing the "window size" of tags that pass through the state.

5.5 Copy State Forward

The *copy state forward* operation is similar to the *copy tag forward* operation; but rather than redirecting incoming edges based on what output tags they generate, it redirects incoming edges based on what state they originate from. This modification operation allows the FST to encode information about the history of states in the transformational FST as part of the model structure.

5.6 Copy Feature Forward

The *copy feature forward* operation is similar to the *copy tag forward* operation; but rather than redirecting incoming edges based on what output tags they generate, it redirects incoming edges based on a feature of the current input value. This modification operation allows the transformation to subdivide output tags based on features of the input value.

6 Hill Climbing System

Having defined a search space, a set of transformations to explore that space, and an evaluation metric, we can use a hill-climbing system to search for a good model structure. This approach starts with a simple initial FST, and makes incremental local changes to that FST until a locally optimal FST is found. In order to help avoid sub-optimal local maxima, we use a fixed-size beam search. To increase the search speed, we used a 12-machine cluster to evaluate candidate FSTs in parallel. The hill climbing system iteratively performs the following procedure:

1. Initialize `candidates` to be the singleton set containing the identity transducer.

2. Repeat ...

   (a) Generate a new FST, by applying a random modification operation to a randomly selected member of the `candidates` set.

   (b) Evaluate the new FSTs, and test its performance on the held-out data set. (This is done in parallel.)

   (c) Once the FST has been evaluated, add it to the `candidates` set.

   (d) Sort the `candidates` set by their score on the held-out data, and discard all but the 10 highest-scoring candidates.

   ... until no improvement is made for twenty consecutive iterations.

3. Return the candidate FST with the highest score.

7 Noun Phrase Chunking Experiment

In order to test this approach to finding a good model structure, we applied our hill-climbing system to the task of noun phrase chunking. The base system was a Linear Chain CRF, implemented using Mallet (McCallum, 2002). The set of features used are listed in Figure 1. Training and testing were performed using the noun phrase chunking corpus described in Ramshaw & Marcus (1995) (Ramshaw and Marcus, 1995). A randomly selected 10% of the original training corpus was used as held-out data, to provide feedback to the hill-climbing system.

7.1 NP Chunking Experiment: Results

Over 100 iterations, the hill-climbing system increased chunking performance on the held-out data from a F-score of 94.93 to an F-score of 95.32. This increase was reflected in an improvement on the test data from an F-score of 92.48 to an F-score

Feature	Description
y_i	The current output tag.
y_i, w_{i+n}	A tuple of the current output tag and the $i+n$th word, $-2 \leq n \leq 2$.
y_i, w_i, w_{i-1}	A tuple of the current output tag, the current word, and the previous word.
y_i, w_i, w_{i+1}	A tuple of the current output tag, the current word, and the next word.
y_i, t_{i+n}	A tuple of the current output tag and the part of speech tag of the $i+n$th word, $-2 \leq n \leq 2$.
$y_i, t_{i+n},$ t_{i+n+1}	A tuple of the current output tag and the two consecutive part of speech tags starting at word $i+n$, $-2 \leq n \leq 1$.
$y_{i+n-1}, t_{i+n},$ t_{i+n+1}	A tuple of the current output tag, and three consecutive part of speech tags centered on word $i+n$, $-1 \leq n \leq 1$.

Table 1: **Feature Set for the CRF NP Chunker**. y_i is the i^{th} output tag; w_i is the i^{th} word; and t_i is the part-of-speech tag for the i^{th} word.

System	F_1 (Held-out)	F_1 (Test)
Baseline (first order)	94.93	92.48
Second order	95.14	92.63
Learned structure	95.32	92.80

Table 2: **Results for NP Chunking Experiment**.

of 92.80.[6] As a point of comparison, a simple second order model achieves an intermediate F-score of 92.63 on the test data. Thus, the model learned by the hill-climbing system outperforms both the simple first-order model and the simple second-order model.

Figure 3 shows how the scores of FSTs on held-out data changed as the hill-climbing system ran. Figure 4 shows the search tree explored by the hill-climbing system.

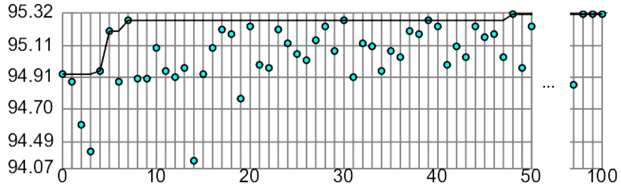


Figure 3: **Performance on Heldout Data for NP Chunking Experiment**. In this graph, each point corresponds to a single transducer generated by the hill-climbing system. The height of each transducer's point indicates its score on held-out data. The line indicates the highest score that has been achieved on the held-out data by any transducer.

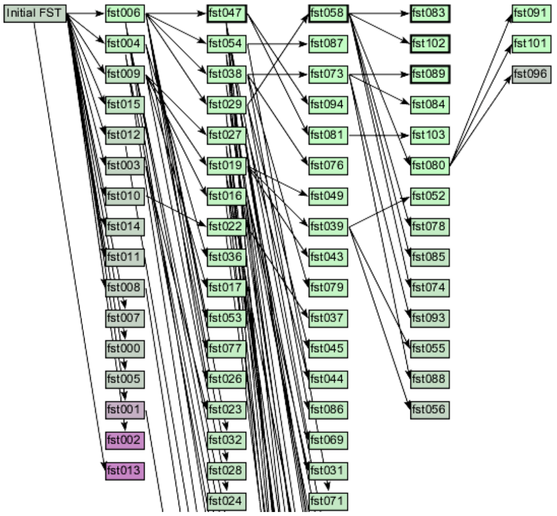


Figure 4: **Hill Climbing Search Tree for NP Chunking Experiment** This tree shows the "ancestry" of each transducer tried by the hill climbing system. Lighter colors indicate higher scores on the held-out data. After one hundred iterations, the five highest scoring transducers were `fst047`, `fst058`, `fst083`, `fst102`, and `fst089`.

[6] The reason that held-out scores are significantly higher than test scores is that held-out data was taken from the same sections of the original corpus as the training data; but test data was taken from new sections. Thus, there was more lexical overlap between the training data and the held-out data than between the training data and the testing data.

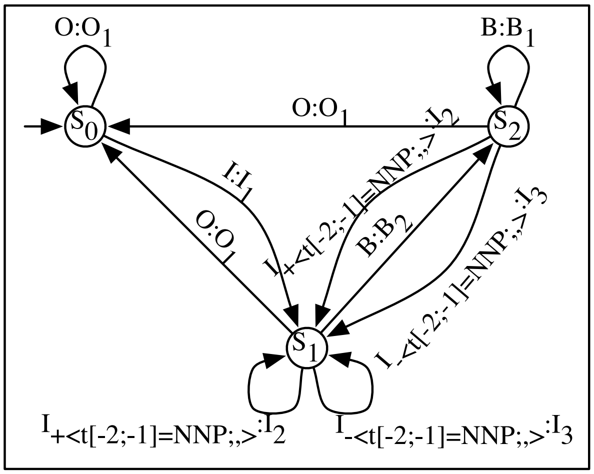


Figure 5: **Final FST**. The highest-scoring FST generated by the hill-climbing algorithm, after a run of 100 iterations. For a discussion of this transducer, see Section 7.1.1.

7.1.1 NP Chunking Experiment: The Selected Transformation

Figure 5 shows the FST for the best output transformation found after 100 iterations of the hill-climbing algorithm. Inspection of this FST reveals that it transforms the original set of three tags (I, O, and B) to six new tags: I_1, I_2, I_3, O, B_1, and B_2.

The first three of these tags are used at the beginning of a chunk: I_1 is used if the preceding tag was O; B_1 is used if the preceding tag was B; and B_2 is used if the preceding tag was I. This is similar to a second order model, in that it records information about both the current tag and the previous tag.

The next tag, O, is used for all words outside of chunks. Thus, the hill-climbing system found that increasing the window size used for O chunks does not help to learn any useful constraints with neighboring tags.

Finally, two tags are used for words that are inside a chunk, but not at the beginning of the chunk: I_2 and I_3. The choice of which tag should be used depends on the input feature that tests whether the current word is a comma, and the previous word was a proper noun (NNP). At first, this might seem like an odd feature to distinguish. But note that in the Wall Street Journal, it is quite common for proper nouns to include internal commas; but for other nouns, it is fairly uncommon. By dividing the I tag in two based on this feature, the model can use separate distributions for these two cases. Thus, the model avoids conflating two contexts that are significantly different from one another for the task at hand.

8 Discussion

Sequential Viterbi Models are capable of learning to model the probability of local patterns on the output structure. But the distance that these patterns can span is limited by the model's structure. This distance can be lengthened by moving to higher order model structures, but only at the expense of an increase in the number of model parameters, along with the data sparsity issues that can arise from that increase. Therefore, it makes sense to be more selective about how we extend the model structure. Using reversible output transformations, it is possible to define model structures that extend the reach of the model only where necessary. And as we have shown here, it is possible to find a suitable output transformation for a given task by using simple search procedures.

9 Acknowledgements

We gratefully acknowledge the support of the National Science Foundation Grant NSF-0415923, Word Sense Disambiguation, the DTO-AQUAINT NBCHC040036 grant under the University of Illinois subcontract to University of Pennsylvania 2003-07911-01, NSF-ITR-0325646: Domain-Independent Semantic Interpretation, and Defense Advanced Research Projects Agency (DARPA/IPTO) under the GALE program, DARPA/CMO Contract No. HR0011-06-C-0022.

References

Ethem Alpaydin, 2004. *Introduction to Machine Learning*, chapter 7. The MIT Press.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598. Morgan Kaufmann, San Francisco, CA.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

José Oncina, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, May.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In David Yarowsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 134–141.

Hong Shen and Anoop Sarkar. 2005. Voting between multiple data representations for text chunking. In *Advances in Artificial Intelligence: 18th Conference of the Canadian Society for Computational Studies of Intelligence*, May.

Andreas Stolcke and Stephen Omohundro. 1993. Hidden Markov Model induction by Bayesian model merging. In C. L. Giles, S. J. Hanson, and J. D. Cowan, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufman, San Mateo, Ca.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.

Erik Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*, Bergen. Association for Computational Linguistics.

Erik Tjong Kim Sang. 2000. Noun phrase recognition by system combination. In *Proceedings of BNAIC*, Tilburg, The Netherlands.

A Statistical Language Modeling Approach to Lattice-Based Spoken Document Retrieval

Tee Kiah Chia[†] **Haizhou Li**[‡] **Hwee Tou Ng**[†]

[†]Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{chiateek,nght}@comp.nus.edu.sg

[‡]Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
hli@i2r.a-star.edu.sg

Abstract

Speech recognition transcripts are far from perfect; they are not of sufficient quality to be useful on their own for spoken document retrieval. This is especially the case for conversational speech. Recent efforts have tried to overcome this issue by using statistics from speech lattices instead of only the 1-best transcripts; however, these efforts have invariably used the classical vector space retrieval model. This paper presents a novel approach to lattice-based spoken document retrieval using statistical language models: a statistical model is estimated for each document, and probabilities derived from the document models are directly used to measure relevance. Experimental results show that the lattice-based language modeling method outperforms both the language modeling retrieval method using only the 1-best transcripts, as well as a recently proposed lattice-based vector space retrieval method.

1 Introduction

Information retrieval (IR) is the task of ranking a collection of documents according to an estimate of their relevance to a query. With the recent growth in the amount of speech recordings in the form of voice mails, news broadcasts, and so forth, the task of spoken document retrieval (SDR) – information retrieval in which the document collection is in the form of speech recordings – is becoming increasingly important.

SDR on broadcast news corpora has been “deemed to be a solved problem”, due to the fact that the performance of retrieval engines working on 1-best automatic speech recognition (ASR) transcripts was found to be “virtually the same as their performance on the human reference transcripts” (NIST, 2000). However, this is still not the case for SDR on data which are more challenging, such as conversational speech in noisy environments, as the 1-best transcripts of these data contain too many recognition errors to be useful for retrieval. One way to ameliorate this problem is to work with not just one ASR hypothesis for each utterance, but multiple hypotheses presented in a *lattice* data structure. A lattice is a connected directed acyclic graph in which each edge is labeled with a term hypothesis and a likelihood value (James, 1995); each path through a lattice gives a hypothesis of the sequence of terms spoken in the utterance.

Each lattice can be viewed as a statistical model of the possible transcripts of an utterance (given the speech recognizer’s state of knowledge); thus, an IR model based on statistical inference will seem to be a more natural and more principled approach to lattice-based SDR. This paper thus proposes a lattice-based SDR method based on the statistical language modeling approach of Song and Croft (1999). In this method, the *expected word count* – the mean number of occurrences of a word given a lattice’s statistical model – is computed for each word in each lattice. Using these expected counts, a statistical language model is estimated for each spoken document, and a document’s relevance to a query is computed as a probability under this model.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 810–818, Prague, June 2007. ©2007 Association for Computational Linguistics

The rest of this paper is organized as follows. In Section 2 we review related work in the areas of speech processing and IR. Section 3 describes our proposed method as well as the baseline methods. Details of the experimental setup are given in Section 4, and experimental results are in Section 5. Finally, Section 6 concludes our discussions and outlines our future work.

2 Related Work

2.1 Lattices for Spoken Document Retrieval

James and Young (1994) first introduced the lattice as a representation for indexing spoken documents, as part of a method for vocabulary-independent keyword spotting. The lattice representation was later applied to the task of spoken document retrieval by James (1995): James counted how many times each query word occurred in each phone lattice with a sufficiently high normalized log likelihood, and these counts were then used in retrieval under a vector space model with $tf \cdot idf$ weighting. Jones et al. (1996) combined retrieval from phone lattices using variations of James' method with retrieval from 1-best word transcripts to achieve better results.

Since then, a number of different methods for SDR using lattices have been proposed. For instance, Siegler (1999) used word lattices instead of phone lattices as the basis of retrieval, and generalized the $tf \cdot idf$ formalism to allow uncertainty in word counts. Chelba and Acero (2005) preprocessed lattices into more compact Position Specific Posterior Lattices (PSPL), and computed an aggregate score for each document based on the posterior probability of edges and the proximity of search terms in the document. Mamou et al. (2006) converted each lattice into a word confusion network (Mangu et al., 2000), and estimated the inverse document frequency (idf) of each word t as the ratio of the total number of words in the document collection to the total number of occurrences of t.

Despite the differences in the details, the above lattice-based SDR methods have all been based on the classical vector space retrieval model with $tf \cdot idf$ weighting.

2.2 Expected Counts from Lattices

A speech recognizer generates a 1-best transcript of a spoken document by considering possible transcripts of the document, and then selecting the transcript with the highest probability. However, unlike a text document, such a 1-best transcript is likely to be inexact due to speech recognition errors. To represent the uncertainty in speech recognition, and to incorporate information from multiple transcription hypotheses rather than only the 1-best, it is desirable to use expected word counts from lattices output by a speech recognizer.

In the context of spoken document search, Siegler (1999) described expected word counts and formulated a way to estimate expected word counts from lattices based on the relative ranks of word hypothesis probabilities; Chelba and Acero (2005) used a more explicit formula for computing word counts based on summing edge posterior probabilities in lattices; Saraclar and Sproat (2004) performed word-spotting in speech lattices by looking for word occurrences whose expected counts were above a certain threshold; and Yu et al. (2005) searched for phrases in spoken documents using a similar measure, the expected word relevance.

Expected counts have also been used to summarize the phonotactics of a speech recording represented in a lattice: Hatch et al. (2005) performed speaker recognition by computing the expected counts of phone bigrams in a phone lattice, and estimating an unsmoothed probability distribution of phone bigrams.

Although many uses of expected counts have been studied, the use of statistical language models built from expected word counts has not been well explored.

2.3 Retrieval via Statistical Language Modeling

Finally, the statistical language modeling approach to retrieval was used by Ponte and Croft (1998) for IR with text documents, and it was shown to outperform the $tf \cdot idf$ approach for this task; this method was further improved on in Song and Croft (1999). Chen et al. (2004) applied Song and Croft's method to Mandarin spoken document retrieval using 1-best ASR transcripts. In this task, it was also shown to outperform $tf \cdot idf$. Thus, the statistical language modeling approach to retrieval has been shown to be superior to the vector space approach for both these IR tasks.

2.4 Contributions of Our Work

The main contributions of our work include

- extending the language modeling IR approach from text-based retrieval to lattice-based spoken document retrieval; and
- formulating a method for building a statistical language model based on expected word counts derived from lattices.

Our method is motivated by the success of the statistical retrieval framework over the vector space approach with $tf \cdot idf$ for text-based IR, as well as for spoken document retrieval via 1-best transcripts. Our use of expected counts differs from Saraclar and Sproat (2004) in that we estimate probability models from the expected counts. Conceptually, our method is close to that of Hatch et al. (2005), as both methods build a language model to summarize the content of a spoken document represented in a lattice. In practice, our method differs from Hatch et al. (2005)'s in many ways: first, we derive word statistics for representing semantics, instead of phone bigram statistics for representing phonotactics; second, we introduce a smoothing mechanism (Zhai and Lafferty, 2004) to the language model that is specific for information retrieval.

3 Methods

We now describe the formulation of three different SDR methods: a baseline statistical retrieval method which works on 1-best transcripts, our proposed statistical lattice-based SDR method, as well as a previously published vector space lattice-based SDR method.

3.1 Baseline Statistical Retrieval Method

Our baseline retrieval method is motivated by Song and Croft (1999), and uses the language model smoothing methods of Zhai and Lafferty (2004). This method is used to perform retrieval on the documents' 1-best ASR transcripts and reference human transcripts.

Let $\mathcal{C}$ be the collection of documents to retrieve from. For each document $\mathbf{d}$ contained in $\mathcal{C}$, and each query $\mathbf{q}$, the relevance of $\mathbf{d}$ to $\mathbf{q}$ can be defined as $\Pr(\mathbf{d} \mid \mathbf{q})$. This probability cannot be computed directly, but under the assumption that the prior $\Pr(\mathbf{d})$ is uniform over all documents in $\mathcal{C}$, we see that

$$\Pr(\mathbf{d} \mid \mathbf{q}) = \frac{\Pr(\mathbf{q} \mid \mathbf{d}) \Pr(\mathbf{d})}{\Pr(\mathbf{q})} \propto \Pr(\mathbf{q} \mid \mathbf{d});$$

This means that ranking documents by $\Pr(\mathbf{d} \mid \mathbf{q})$ is equivalent to ranking them by $\Pr(\mathbf{q} \mid \mathbf{d})$, and thus $\Pr(\mathbf{q} \mid \mathbf{d})$ can be used to measure relevance (Berger and Lafferty, 1999).

Now express $\mathbf{q}$ as a series of words drawn from a vocabulary $\mathcal{V} = \{w_1, w_2, \cdots w_V\}$; that is, $\mathbf{q} = q_1 q_2 \cdots q_K$, where K is the number of words in the query, and $q_i \in \mathcal{V}$ for $1 \leq i \leq K$. Then given a unigram model derived from $\mathbf{d}$ which assigns a probability $\Pr(w \mid \mathbf{d})$ to each word w in $\mathcal{V}$, we can compute $\Pr(\mathbf{q} \mid \mathbf{d})$ as follows:

$$\begin{aligned} \Pr(\mathbf{q} \mid \mathbf{d}) &= \Pr(q_1 q_2 \cdots q_K \mid \mathbf{d}) \\ &= \prod_{i=1}^{K} \Pr(q_i \mid \mathbf{d}) \\ &= \prod_{\substack{w \in \mathcal{V}, \\ C(w|\mathbf{q})>0}} \Pr(w|\mathbf{d})^{C(w|\mathbf{q})} \qquad (1) \end{aligned}$$

where $C(w \mid \mathbf{q})$ is the word count of w in $\mathbf{q}$.

Before using Equation 1, we must estimate a unigram model from $\mathbf{d}$: that is, an assignment of probabilities $\Pr(w \mid \mathbf{d})$ for all $w \in \mathcal{V}$. One way to do this is to use a maximum likelihood estimate (MLE) – an assignment of $\Pr(w \mid \mathbf{d})$ for all w which maximizes the probability of generating $\mathbf{d}$. The MLE is given by the equation

$$\Pr{}_{\text{mle}}(w \mid \mathbf{d}) = \frac{C(w \mid \mathbf{d})}{|\mathbf{d}|}$$

where $C(w \mid \mathbf{d})$ is the number of occurrences of w in $\mathbf{d}$, and $|\mathbf{d}|$ is the total number of words in $\mathbf{d}$. However, using this formula means we will get a value of zero for $\Pr(\mathbf{q} \mid \mathbf{d})$ if even a single query word q_i is not found in $\mathbf{d}$. To overcome this problem, we smooth the model by assigning some probability mass to such unseen words. Specifically, we adopt a two-stage smoothing method (Zhai and Lafferty, 2004):

$$\begin{aligned}\Pr(w \mid \mathbf{d}) &= (1-\lambda)\frac{C(w \mid \mathbf{d}) + \mu \Pr(w \mid \mathcal{C})}{|\mathbf{d}| + \mu} \\ &\quad + \lambda \Pr(w \mid \mathcal{U}) \qquad (2)\end{aligned}$$

Here, $\mathcal{U}$ denotes a background language model, and $\mu > 0$ and $\lambda \in (0,\, 1)$ are parameters to the smoothing procedure. This is a combination of Bayesian smoothing using Dirichlet priors (MacKay and Peto, 1984) and Jelinek-Mercer smoothing (Jelinek and Mercer, 1980).

The parameter λ can be set empirically according to the nature of the queries. For the parameter μ, we adopt the estimation procedure of Zhai and Lafferty (2004): we maximize the leave-one-out log likelihood of the document collection, namely

$$\begin{aligned}\ell_{-1}(\mu \mid \mathcal{C}) = \sum_{\mathbf{d}\in\mathcal{C}} \sum_{w\in\mathcal{V}} C(w \mid \mathbf{d}) \\ \log\left(\frac{C(w \mid \mathbf{d}) - 1 + \mu \Pr(w \mid \mathcal{C})}{|\mathbf{d}| - 1 + \mu}\right) \qquad (3)\end{aligned}$$

by using Newton's method to solve the equation

$$\ell'_{-1}(\mu \mid \mathcal{C}) = 0$$

3.2 Our Proposed Statistical Lattice-Based Retrieval Method

We now propose our lattice-based retrieval method. In contrast to the above baseline method, our proposed method works on the lattice representation of spoken documents, as generated by a speech recognizer.

First, each spoken document is divided into M short speech segments. A speech recognizer then generates a lattice for each speech segment. As previously stated, a lattice is a connected directed acyclic graph with edges labeled with word hypotheses and likelihoods. Thus, each path through the lattice contains a hypothesis of the series of words spoken in this speech segment, $\mathbf{t} = t_1 t_2 \cdots t_N$, along with acoustic probabilities $\Pr(o_1 \mid t_1)$, $\Pr(o_2 \mid t_2)$, $\cdots$ $\Pr(o_N \mid t_N)$, where o_i denotes the acoustic observations for the time interval of the word t_i hypothesized by the speech recognizer. Let $\mathbf{o} = o_1 o_2 \cdots o_N$ denote the acoustic observations for the entire speech segment; then

$$\Pr(\mathbf{o} \mid \mathbf{t}) = \prod_{i=1}^{N} \Pr(o_i \mid t_i)$$

We then rescore each lattice with an n-gram language model. Effectively, this means multiplying the acoustic probabilities with n-gram probabilities:

$$\begin{aligned}\Pr(\mathbf{t}, \mathbf{o}) &= \Pr(\mathbf{o} \mid \mathbf{t}) \Pr(\mathbf{t}) \\ &= \prod_{i=1}^{N} \Pr(o_i \mid t_i) \Pr(t_i \mid t_{i-n+1} \cdots t_{i-1})\end{aligned}$$

This produces an expanded lattice in which paths (hypotheses) are weighted by their posterior probabilities rather than their acoustic likelihoods: specifically, by $\Pr(\mathbf{t}, \mathbf{o}) \propto \Pr(\mathbf{t} \mid \mathbf{o})$ rather than $\Pr(\mathbf{o} \mid \mathbf{t})$ (Odell, 1995). The lattice is then pruned, by removing those paths in the lattice whose log posterior probabilities – to be precise, whose $\gamma \ln \Pr(\mathbf{t} \mid \mathbf{o})$ – are not within a threshold Θ of the best path's log posterior probability (in our implementation, $\gamma = 10000.5$).

Next, we compute the expected count of each word in each document. For each word w and each document $\mathbf{d}$ comprised of M speech segments represented by M acoustic observations $\mathbf{o}^{(1)}$, $\mathbf{o}^{(2)}$, $\cdots$ $\mathbf{o}^{(M)}$, the expected count of w in $\mathbf{d}$ is

$$\mathrm{E}[C(w \mid \mathbf{d})] = \sum_{j=1}^{M} \sum_{\mathbf{t}} C(w \mid \mathbf{t}) \Pr(\mathbf{t} \mid \mathbf{o}^{(j)})$$

where $C(w \mid \mathbf{t})$ is the word count of w in the hypothesized transcript $\mathbf{t}$. We can also analogously compute the expected document length:

$$\mathrm{E}[|\mathbf{d}|] = \sum_{j=1}^{M} \sum_{\mathbf{t}} |\mathbf{t}| \Pr(\mathbf{t} \mid \mathbf{o}^{(j)})$$

where $|\mathbf{t}|$ denotes the number of words in $\mathbf{t}$.

We now replace $C(w \mid \mathbf{d})$ and $|\mathbf{d}|$ in Equation 2 with $\mathrm{E}[C(w \mid \mathbf{d})]$ and $\mathrm{E}[|\mathbf{d}|]$; thus

$$\begin{aligned}\Pr(w \mid \mathbf{d}) &= (1-\lambda)\frac{\mathrm{E}[C(w \mid \mathbf{d})] + \mu \Pr(w \mid \mathcal{C})}{\mathrm{E}[|\mathbf{d}|] + \mu} \\ &\quad + \lambda \Pr(w \mid \mathcal{U}) \qquad (4)\end{aligned}$$

In addition, we also modify the procedure for estimating μ, by replacing $C(w \mid \mathbf{d})$ and

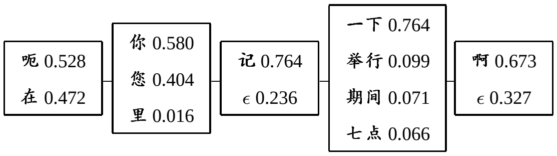


Figure 1: Example of a word confusion network

$|\mathbf{d}|$ in Equation 3 with $\left\lfloor \mathrm{E}[C(w \mid \mathbf{d})] + \frac{1}{2} \right\rfloor$ and $\sum_{w \in \mathcal{V}} \left\lfloor \mathrm{E}[C(w \mid \mathbf{d})] + \frac{1}{2} \right\rfloor$ respectively. The probability estimates from Equation 4 can then be substituted into Equation 1 to yield relevance scores.

3.3 Baseline $tf \cdot idf$ Lattice-Based Retrieval Method

As a further comparison, we also implemented Mamou et al. (2006)'s vector space retrieval method (without query refinement via lexical affinities). In this method, each document $\mathbf{d}$ is represented as a word confusion network (WCN) (Mangu et al., 2000) – a simplified lattice which can be viewed as a sequence of confusion sets $c_1, c_2, c_3, \cdots$. Each c_i corresponds approximately to a time interval in the spoken document and contains a group of word hypotheses, and each word w in this group of hypotheses is labeled with the probability $\Pr(w \mid c_i, \mathbf{d})$ – the probability that w was spoken in the time interval of c_i. A confusion set may also give a probability for $\Pr(\epsilon \mid c_i, \mathbf{d})$, the probability that no word was spoken in the time of c_i. Figure 1 gives an example of a WCN.

Mamou et al.'s retrieval method proceeds as follows. First, the documents are divided into speech segments, lattices are generated from the speech segments, and the lattices are pruned according to the path probability threshold Θ, as described in Section 3.2. The lattice for each speech segment is then converted into a WCN according to the algorithm of Mangu et al. (2000). The WCNs for the speech segments in each document are then concatenated to form a single WCN per document.

Now, to retrieve documents in response to a query $\mathbf{q}$, the method computes, for each document $\mathbf{d} \in \mathcal{C}$ and each word $w \in \mathcal{V}$,

- the "document length" $|\mathbf{d}|$, computed as the number of confusion sets in the WCN of $\mathbf{d}$;
- the "average document length" $avdl$, computed as

$$avdl \quad = \quad \frac{1}{|\mathcal{C}|} \sum_{\mathbf{d}' \in \mathcal{C}} |\mathbf{d}'| \, ;$$

- the "document term frequency" $C^*(w \mid \mathbf{d})$, computed as

$$C^*(w|\mathbf{d}) = \sum_{c \in occ(w,\mathbf{d})} (b_{rank(w|c,\mathbf{d})} \cdot \Pr(w|c,\mathbf{d}))$$

  where $occ(w, \mathbf{d})$ is the set of confusion sets in $\mathbf{d}$'s WCN which contain w as a hypothesis, $rank(w \mid c, \mathbf{d})$ is the rank of w in terms of probability within the confusion set c, and $(b_1, b_2, b_3, \cdots) =$ (10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 0, 0, $\cdots$) is a boosting vector which serves to discard all but the top 10 hypotheses, and gives more weight to higher-ranked word hypotheses;

- the query term frequency $C(w \mid \mathbf{q})$, which is simply the word count of w in $\mathbf{q}$; and
- the "inverse document frequency" $idf(w)$, computed as

$$idf(w) \quad = \quad \log \frac{O}{O_w}$$

  where

$$O_w \quad = \quad \sum_{\mathbf{d} \in \mathcal{C}} \sum_{c \in occ(w,\mathbf{d})} \Pr(w \mid c, \mathbf{d})$$

$$O \quad = \quad \sum_{w' \in \mathcal{V}} O_{w'}$$

With these, the relevance of $\mathbf{d}$ to $\mathbf{q}$ is computed as (Carmel et al., 2001)

$$rel(\mathbf{d}, \mathbf{q}) \quad = \quad \frac{\sum_{w \in \mathcal{V}} C^*(w \mid \mathbf{d}) \cdot C(w \mid \mathbf{q}) \cdot idf(w)}{\sqrt{0.8 \cdot avdl + 0.2 \cdot |\mathbf{d}|}}$$

4 Experiments

4.1 Document Collection

To evaluate our proposed retrieval method, we performed experiments using the Hub5 Mandarin training corpus released by the Linguistic Data Consortium (LDC98T26). This is a conversational telephone speech corpus which is 17 hours long, and contains recordings of 42 telephone calls corresponding to approximately 600Kb of transcribed Mandarin text. Each conversation has been broken up into speech segments of less than 8 seconds each.

As the telephone calls in LDC98T26 have not been divided neatly into "documents", we had to choose a suitable unit of retrieval which could serve as a "document". An entire conversation would be too long for such a purpose, while a speech segment or speaker turn would be too short. We decided to use $\frac{1}{2}$-minute time windows with 50% overlap as retrieval units, following Abberley et al. (1999) and Tuerk et al. (2001). The 42 telephone conversations were thus divided into 4,312 retrieval units ("documents"). Each document comprises multiple consecutive speech segments.

4.2 Queries and Ground Truth Relevance Judgements

We then formulated 18 queries (14 test queries, 4 development queries) to issue on the document collection. Each query was comprised of one or more written Chinese keywords. We then obtained ground truth relevance judgements by manually examining each of the 4,312 documents to see if it is relevant to the topic of each query. The number of retrieval units relevant to each query was found to range from 4 to 990. The complete list of queries and the number of documents relevant to each query are given in Table 1.

4.3 Preprocessing of Documents and Queries

Next, we processed the document collection with a speech recognizer. For this task we used the Abacus system (Hon et al., 1994), a large vocabulary continuous speech recognizer which contains a triphone-based acoustic system and a frame-synchronized search algorithm for effective word decoding. Each Mandarin syllable was modeled by one to four triphone models. Acoustic models were trained from a corpus of 200 hours of telephony speech from 500 speakers sampled at 8kHz. For each speech frame, we extracted a 39-dimensional feature vector consisting of 12 MFCCs and normalized energy, and their first and second order derivatives. Sentence-based cepstral mean subtraction was applied for acoustic normalization both in the training and testing. Each triphone was modeled by a left-to-right 3-state hidden Markov model (HMM), each state having 16 Gaussian mixture components. In total, we built 1,923 untied within-syllable triphone models for 43 Mandarin phonemes, as well as 3 silence models. The search algorithm was supported by a loop grammar of over 80,000 words.

Test queries		
Topic	Keywords	# relevant documents
Contact information	电话, 号码, 地址, 联系, 姓	103
Chicago	芝加哥	15
The weather	天气, 冷, 热, 暖和, 风, 凉快, 雨, 空调, 干燥, 潮湿, 气候, 温度	117
Housing matters	房子, 家, 住, 房租, 家具, 搬, 厨房, 卧室, 水电, 房东, 院子	354
Studies, academia	毕业, 学位, 考试, 修, 读, 托福, 念书, 课, 学分, 进修, 学费, 同学	990
Litigation	法律, 律师, 打官司, 起诉	31
Raising children	小孩, 孩子, 生育, 儿子, 幼儿园, 玩, 玩具, 女儿	334
Christian churches	教会, 神, 主, 礼拜, 教堂, 活动, 圣经, 团契	78
Floods	洪水, 淹, 堤, 水	4
Clothing	衣服, 皮衣, 帽子, 裤子, 高统袜, 袜子, 牛仔裤, 西服, 穿	28
Eating out	吃, 餐馆, 外卖, 中餐, 请客, 饭店	57
Playing sports	打球, 活动, 橄榄球, 排球	24
Dealings with banks	银行, 支票, 钱, 开户, 贷款	54
Computers and software	电脑, 计算机, 软件	175

Development queries		
Topic	Keywords	# relevant documents
Passport and visa matters	护照, 签证, 入境, 手续, 绿卡, 移民	143
Washington D. C.	华盛顿	15
Working life	活儿, 钱, 打工, 税, 工作, 老板, 出差, 公司, 挣, 工资, 上司, 同事, 忙, 职业	509
1996 Olympics	奥运会, 亚特兰大	8

Table 1: List of test and development queries

We processed the speech segments in our collection corpus, to generate lattices incorporating acoustic likelihoods but not n-gram model probabilities. We then rescored the lattices using a backoff trigram language model interpolated in equal proportions from two trigram models:

- a model built from the TDT-2, TDT-3, and TDT-4 Mandarin news broadcast transcripts (about 58Mb of text)
- a model built from corpora of transcripts of conversations, comprised of a 320Kb subset of the Callhome Mandarin corpus (LDC96T16) and the CSTSC-Flight corpus from the Chinese Corpus Consortium (950Kb)

The unigram counts from this model were also used as the background language model $\mathcal{U}$ in Equations 2 and 4.

The reference transcripts, queries, and trigram model training data were all segmented into words using Low et al. (2005)'s Chinese word segmenter, trained on the Microsoft Research (MSR) corpus, with the speech recognizer's vocabulary used as an external dictionary. The 1-best ASR transcripts were decoded from the rescored lattices.

Lattice rescoring, trigram model building, WCN generation, and computation of expected word counts were done using the SRILM toolkit (Stolcke, 2002), while lattice pruning was done with the help of the AT&T FSM Library (Mohri et al., 1998).

We also computed the character error rate (CER) and syllable error rate (SER) of the 1-best transcripts, and the lattice oracle CER, for one of the telephone conversations in the speech corpus (`ma_4160`). The CER was found to be 69%, the SER 63%, and the oracle CER 29%.

4.4 Retrieval and Evaluation

We then performed retrieval on the document collection using the algorithms in Section 3, using the reference transcripts, the 1-best ASR transcripts, lattices, and WCNs. We set $\lambda = 0.1$, which was suggested by Zhai and Lafferty (2004) to give good retrieval performance for keyword queries.

The results of retrieval were checked against the ground truth relevance judgements, and evaluated in terms of the non-interpolated mean average precision (MAP):

$$\text{MAP} = \frac{1}{L}\sum_{i=1}^{L}\left(\frac{1}{R_i}\sum_{j=1}^{R_i}\frac{j}{r_{i,j}}\right)$$

where L denotes the total number of queries, R_i the total number of documents relevant to the ith query, and $r_{i,j}$ the position of the jth relevant document in the ranked list output by the retrieval method for query i.

For the lattice-based retrieval methods, we performed retrieval with the development queries using several values of Θ between 0 and 100,000, and then used the value of Θ with the best MAP to do retrieval with the test queries.

Retrieval method	Retrieval source	MAP for development queries	MAP for test queries
Statistical	Reference transcripts	0.5052	0.4798
Statistical	1-best transcripts	0.1251	0.1364
Vector space $tf \cdot idf$	Lattices, $\Theta = 27,500$	0.1685	0.1599
Statistical	Lattices, $\Theta = 65,000$	0.2180	0.2154

Table 2: Summary of experimental results

5 Experimental Results

The results of our experiments are summarized in Table 2; the MAP of the two lattice-based retrieval methods, Mamou et al. (2006)'s vector space method and our proposed statistical retrieval method, are shown in Figure 2 and Figure 3 respectively.

The results show that, for the vector space retrieval method, the MAP of the development queries is highest at $\Theta = 27,500$, at which point the MAP for the test queries is 0.1599; and for our proposed method, the MAP for the development queries is highest at $\Theta = 65,000$, and at this point the MAP for the test queries reaches 0.2154.

As can be seen, the performance of our statistical lattice-based method shows a marked improvement over the MAP of 0.1364 achieved using only the 1-best ASR transcripts, and indeed a one-tailed Student's t-test shows that this improvement is statistically significant at the 99.5% confidence level. The statistical method also yields better performance than Mamou et al.'s vector space method – a t-test

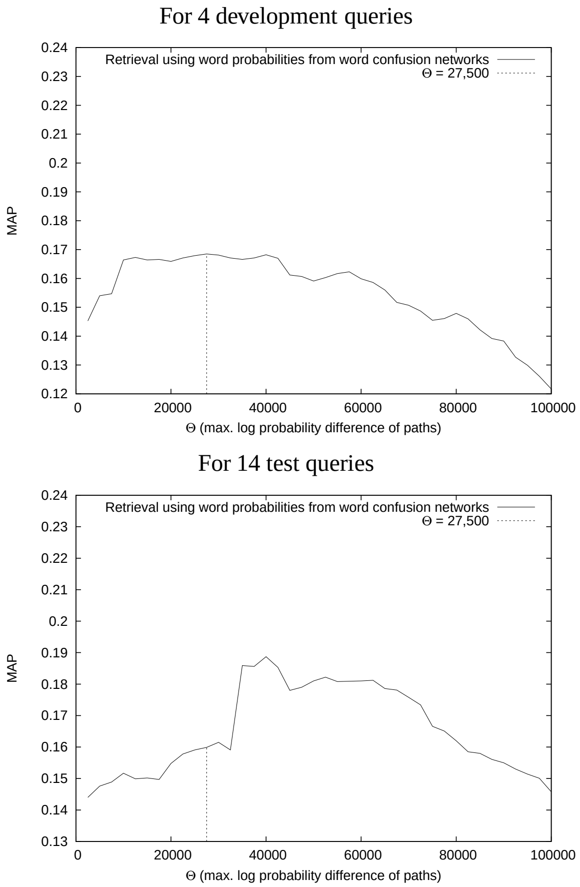


Figure 2: MAP of Mamou et al. (2006)'s vector space method for lattice-based retrieval, at various pruning thresholds Θ

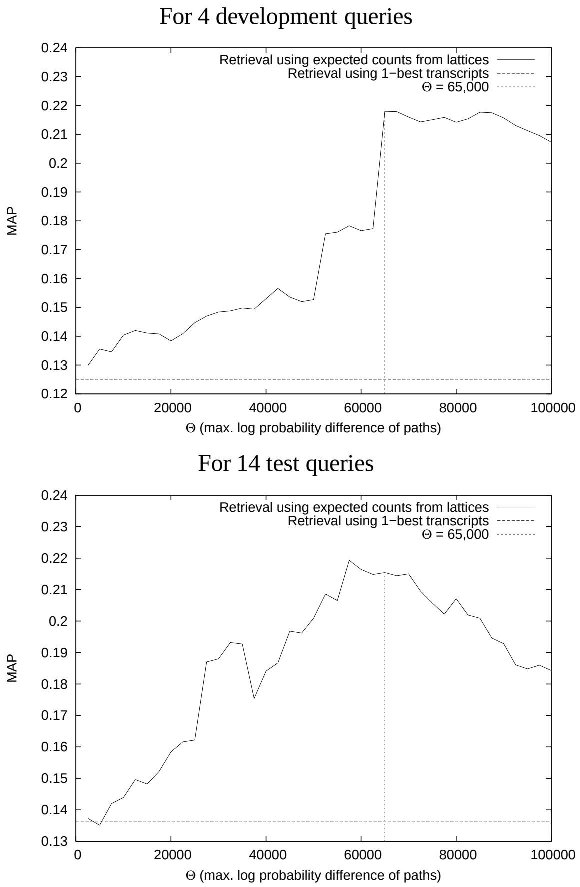


Figure 3: MAP of our proposed statistical method for lattice-based retrieval, at various pruning thresholds Θ

shows the performance difference to be statistically significant at the 97.5% confidence level.

6 Conclusions and Future Work

We have presented a method for performing spoken document retrieval using lattices which is based on a statistical language modeling retrieval framework. Results show that our new method can significantly improve the retrieval MAP compared to using only the 1-best ASR transcripts. Also, our proposed retrieval method has been shown to outperform Mamou et al. (2006)'s vector space lattice-based retrieval method.

Besides the better empirical performance, our method also has other advantages over Mamou et al.'s vector space method. For one, our method computes expected word counts directly from rescored lattices, and does not require an additional step to convert lattices lossily to WCNs. Furthermore, our method uses all the hypotheses in each lattice, rather than just the top 10 word hypotheses at each time interval. Most importantly, our method provides a more natural and more principled approach to lattice-based spoken document retrieval based on a sound statistical foundation, by harnessing the fact that lattices are themselves statistical models; the statistical approach also means that our method can be more easily augmented with additional statistical knowledge sources in a principled way.

For future work, we plan to test our proposed method on English speech corpora, and with larger-scale retrieval tasks involving more queries and more documents. We would like to extend our method to other speech processing tasks, such as spoken document classification and example-based spoken document retrieval as well.

References

Dave Abberley, David Kirby, Steve Renals, and Tony Robinson. 1999. The THISL broadcast news retrieval system. In *Proceedings of ESCA ETRW Workshop on Accessing Information in Spoken Audio*, pages 14–19.

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of SIGIR 1999*, pages 222–229.

David Carmel, Einat Amitay, Miki Herscovici, Yoelle Maarek, Yael Petruschka, and Aya Soffer. 2001. Juru at TREC 10 – Experiments with index pruning. In *Proceedings of the Tenth Text Retrieval Conference (TREC-10)*, pages 228–236.

Ciprian Chelba and Alex Acero. 2005. Position specific posterior lattices for indexing speech. In *Proceedings of ACL 2005*, pages 443–450.

Berlin Chen, Hsin-min Wang, and Lin-shan Lee. 2004. A discriminative HMM/n-gram-based retrieval approach for Mandarin spoken documents. *ACM Transactions on Asian Language Information Processing*, 3(2):128–145.

Andrew O. Hatch, Barbara Peskin, and Andreas Stolcke. 2005. Improved phonetic speaker recognition using lattice decoding. In *Proceedings of IEEE ICASSP 2005*, 1:169–172.

Hsiao-Wuen Hon, Baosheng Yuan, Yen-Lu Chow, S. Narayan, and Kai-Fu Lee. 1994. Towards large vocabulary Mandarin Chinese speech recognition. In *Proceedings of IEEE ICASSP 1994*, 1:545–548.

David Anthony James and Steve J. Young. 1994. A fast lattice-based approach to vocabulary independent wordspotting. In *Proceedings of ICASSP 1994*, 1:377–380.

David Anthony James. 1995. *The Application of Classical Information Retrieval Techniques to Spoken Documents.* Ph. D. thesis, University of Cambridge.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397.

Gareth J. F. Jones, Jonathan T. Foote, Karen Spärck Jones, and Steve J. Young. 1996. Retrieving spoken documents by combining multiple index sources. In *Proceedings of SIGIR 1996*, pages 30–38.

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 161–164.

David J. C. MacKay and Linda C. Bauman Peto. 1994, A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):1–19.

Jonathan Mamou, David Carmel, and Ron Hoory. 2006. Spoken document retrieval from call-center conversations. In *Proceedings of SIGIR 2006*, pages 51–58.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 1998. A rational design for a weighted finite-state transducer library. *Lecture Notes in Computer Science*, 1436:144–158.

National Institute of Standards and Technology. 2000. TREC-9 SDR Track web site. www.nist.gov/speech/tests/sdr/sdr2000/sdr2000.htm.

Julian James Odell. 1995. *The Use of Context in Large Vocabulary Speech Recognition*. Ph. D. thesis, Cambridge University Engineering Department.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR 1998*, pages 275–281.

Murat Saraclar and Richard Sproat. 2004. Lattice-based search for spoken utterance retrieval. In *Proceedings of HLT-NAACL 2004*, pages 129–136.

Matthew A. Siegler. 1999. *Integration of Continuous Speech Recognition and Information Retrieval for Mutually Optimal Performance.* Ph. D. thesis, Carnegie Mellon University.

Fei Song and W. Bruce Croft. 1999. A general language model for information retrieval. In *Proceedings of CIKM 1999*, pages 316–321.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of ICSLP*, 2:901–904.

Andy Tuerk, Sue E. Johnson, Pierre Jourlin, Karen Spärck Jones, and Philip C. Woodland. 2001. The Cambridge University multimedia document retrieval demo system. *International Journal of Speech Technology*, 4(3–4):241–250.

Peng Yu, Kaijiang Chen, Lie Lu, and Frank Seide. 2005. Searching the audio notebook: keyword search in recorded conversations. In *Proceedings of HLT/EMNLP 2005*, pages 947–954.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214.

Learning Noun Phrase Query Segmentation

Shane Bergsma and Qin Iris Wang
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{bergsma,wqin}@cs.ualberta.ca

Abstract

Query segmentation is the process of taking a user's search-engine query and dividing the tokens into individual phrases or semantic units. Identification of these query segments can potentially improve both document-retrieval precision, by first returning pages which contain the exact query segments, and document-retrieval recall, by allowing query expansion or substitution via the segmented units. We train and evaluate a machine-learned query segmentation system that achieves 86% segmentation-decision accuracy on a gold standard set of segmented noun phrase queries, well above recently published approaches. Key enablers of this high performance are features derived from previous natural language processing work in noun compound bracketing. For example, token association features beyond simple N-gram counts provide powerful indicators of segmentation.

1 Introduction

Billions of times every day, people around the world communicate with Internet search engines via a small text box on a web page. The user provides a sequence of words to the search engine, and the search engine interprets the query and tries to return web pages that not only *contain* the query tokens, but that are also somehow *about* the topic or idea that the query terms describe.

Recent years have seen a widespread recognition that the user is indeed providing natural language text to the search engine; query tokens are not independent, unordered symbols to be matched on a web document but rather ordered words and phrases with syntactic relationships. For example, Zhai (1997) pointed out that indexing on single-word symbols is not able to distinguish a search for "bank terminology" from one for "terminology bank." The reader can submit these queries to a current search engine to confirm that modern indexing does recognize the effect of token order on query meaning in some way.

Accurately interpreting query semantics also depends on establishing relationships between the query tokens. For example, consider the query "two man power saw." There are a number of possible interpretations of this query, and these can be expressed through a number of different segmentations or bracketings of the query terms:

1. [two man power saw]
2. [two man] [power saw]
3. [two] [man] [power saw]
4. [two] [man power] [saw], etc.

One simple way to make use of these interpretations in search would be to put quotation marks around the phrasal segments to require the search engine to only find pages with exact phrase matches. If, as seems likely, the searcher is seeking pages about the large, mechanically-powered two-man saws used by lumberjacks and sawyers to cut big trees, then the first segmentation is correct. Indeed, a phrasal search for "two man power saw" on Google does find the device of interest. So does the second interpretation, but along with other, less-relevant pages discussing competitions involving "*two-man* handsaw,

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 819–826, Prague, June 2007. ©2007 Association for Computational Linguistics

two-woman handsaw, *power saw* log bucking, etc." The top document returned for the third interpretation, meanwhile, describes a *man* on a rampage at a subway station with *two* cordless *power saws*, while the fourth interpretation finds pages about topics ranging from hockey's thrilling *two-man power* play advantage to the *man power* situation during the Second World War. Clearly, choosing the right segmentation means finding the right documents faster.

Query segmentation can also help if insufficient pages are returned for the original query. A technique such as query substitution or expansion (Jones et al., 2006) can be employed using the segmented units. For example, we could replace the sexist "two man" modifier with the politically-correct "two person" phrase in order to find additional relevant documents. Without segmentation, expanding via the individual words "two," "man," "power," or "saw" could produce less sensible results.

In this paper, we propose a data-driven, machine-learned approach to query segmentation. Similar to previous segmentation approaches described in Section 2, we make a decision to segment or not to segment between each pair of tokens in the query. Unlike previous work, we view this as a classification task where the decision parameters are learned discriminatively from gold standard data. In Section 3, we describe our approach and the features we use. Section 4 describes our labelled data, as well as the specific tools used for our experiments. Section 5 provides the results of our evaluation, and shows the strong gains in performance possible using a wide set of features within a discriminative framework.

2 Related Work

Query segmentation has previously been approached in an unsupervised manner. Risvik et al. (2003) combine the frequency count of a segment and the mutual information (MI) between pairs of words in the segment in a heuristic scoring function. The system chooses the segmentation with the highest score as the output segmentation. Jones et al. (2006) use MI between pairs of tokens as the sole factor in deciding on segmentation breaks. If the MI is above a threshold (optimized on a small training set), the pair of tokens is joined in a segment. Otherwise, a segmentation break is made.

Query segmentation is related to the task of noun compound (NC) bracketing. NC bracketing determines the syntactic structure of an NC as expressed by a binary tree, or, equivalently, a binary bracketing (Nakov and Hearst, 2005a). Zhai (1997) first identified the importance of syntactic query/corpus parsing for information retrieval, but did not consider query segmentation itself. In principle, as N increases, the number of binary trees for an N-token compound is much greater than the 2^{N-1} possible segmentations. In practice, empirical NC research has focused on three-word compounds. The computational problem is thus deciding whether the three-word NC has a left or right-bracketing structure (Lauer, 1995). For the segmentation task, analysing a three-word NC requires deciding between four different segmentations. For example, there are two bracketings for "used car parts," the left-bracketing "[[used car] parts]" and the right-bracketing "[used [car parts]]," while there are four segmentations, including the case where there is only one segment, "[used car parts]" and the base case where each token forms its own segment, "[used] [car] [parts]." Query segmentation thus naturally handles the case where the query consists of multiple, separate noun phrases that should not be analysed with a single binary tree.

Despite the differences between the tasks, it is worth investigating whether the information that helps disambiguate left and right-bracketings can also be useful for segmentation. In particular, we explored many of the sources of information used by Nakov and Hearst (2005a), as well as several novel features that aid segmentation performance and should also prove useful for NC analysis researchers. Unlike all previous approaches that we are aware of, we apply our features in a flexible discriminative framework rather than a classification based on a vote or average of features.

NC analysis has benefited from the recent trend of using web-derived features rather than corpus-based counts (Keller and Lapata, 2003). Lapata and Keller (2004) first used web-based co-occurrence counts for the bracketing of NCs. Recent innovations have been to use statistics "beyond the N-gram," such as counting the number of web pages where a pair of words w, x participate in a genitive relationship ("w's x"), occur collapsed as a single phrase ("wx") (Nakov and Hearst, 2005a) or have a definite article as a left-boundary marker ("the w x") (Nicholson and Baldwin, 2006). We show strong performance gains when such features are employed for query segmentation.

NC bracketing is part of a larger field of research on multiword expressions including general NC interpretation. NC interpretation explores not just the syntactic dependencies among compound constituents, but the semantics of the nominal relationships (Girju et al., 2005). Web-based statistics have also had an impact on these wider analysis tasks, including work on interpretation of verb nominalisations (Nicholson and Baldwin, 2006) and NC coordination (Nakov and Hearst, 2005b).

3 Methodology

3.1 Segmentation Classification

Consider a query $\boldsymbol{x} = \{x_1, x_2, ..., x_N\}$ consisting of N query tokens. Segmentation is a mapping $S : \boldsymbol{x} \rightarrow \boldsymbol{y} \in Y_N$, where $\boldsymbol{y}$ is a segmentation from the set Y_N. Since we can either have or not have a segmentation break at each of the $N-1$ spaces between the N tokens, $|Y_N| = 2^{N-1}$. Supervised machine learning can be applied to derive the mapping S automatically, given a set of training examples consisting of pairs of queries and their segmentations $T = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}$. Typically this would be done via a set of features $\Psi(\boldsymbol{x}, \boldsymbol{y})$ for the structured examples. A set of weights $\boldsymbol{w}$ can be learned discriminatively such that each training example $(\boldsymbol{x_i}, \boldsymbol{y_i})$ has a higher score, $Score_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{w} \cdot \Psi(\boldsymbol{x}, \boldsymbol{y})$, than alternative query-segmentation pairs, $(\boldsymbol{x_i}, \boldsymbol{z_i}), z_i \neq y_i$.[1] At test time, the classifier chooses the segmentation for $\boldsymbol{x}$ that has the highest score according to the learned parameterization: $\hat{\boldsymbol{y}} = \text{argmax}_{\boldsymbol{y}} Score_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y})$. Unlike many problems in NLP such as parsing or part-of-speech tagging, the small cardinality of Y_N makes enumerating all the alternative query segmentations computationally feasible.

In our preliminary experiments, we used a Support Vector Machine (SVM) ranker (Joachims, 2002) to learn the structured classifier.[2] We also investigated a Hidden Markov Model SVM (Altun et al., 2003) to label the segmentation breaks using information from past segmentation decisions. Ultimately, the mappings produced by these approaches were not as accurate as a simple formulation that creates a full query segmentation $\boldsymbol{y}$ as the combination of independent classification decisions made between each pair of tokens in the query.[3]

In the classification framework, the input is a query, $\boldsymbol{x}$, a position in the query, i, where $0<i<N$, and the output is a segmentation decision yes/no. The training set of segmented queries is converted into examples of decisions between tokens and learning is performed on this set. At test time, $N-1$ segmentation decisions are made for the N-length query and an output segmentation $\boldsymbol{y}$ is produced. Here, features depend only on the input query $\boldsymbol{x}$ and the position in the query i. For a decision at position i, we use features from tokens up to three positions to the left and to the right of the decision location. That is, for a decision between x_{L0} and x_{R0}, we extract features from a window of six tokens in the query: $\{..., x_{L2}, x_{L1}, x_{L0}, x_{R0}, x_{R1}, x_{R2}, ...\}$. We now detail the features derived from this window.

3.2 Features

There are a number of possible indicators of whether a segmentation break occurs between a pair of tokens. Some of these features fire separately for each token x in our feature window, while others are defined over pairs or sets of tokens in the window. We first describe the features that are defined for the tokens around the decision boundary, x_{L0} and x_{R0}, before describing how these same features are extended to longer phrases and other token pairs.

3.2.1 Decision-boundary features

Table 1 lists the binary features that fire if particular aspects of a token or pair of tokens are present. For example, one of the *POS-tags* features will fire if the pair's part-of-speech tags are DT JJ, another feature will fire if the position of the pair in the to-

[1] See e.g. Collins (2002) for a popular training algorithm.

[2] A ranking approach was also used previously by Daumé III and Marcu (2004) for the CoNLL-99 nested noun phrase identification task.

[3] The structured learners did show large gains over the classification framework on the dev-set when using only the basic features for the decision-boundary tokens (see Section 3.2.1), but not when the full feature set was deployed. Also, features only available to structured learners, e.g. number of segments in query, etc., did improve the performance of the structured approaches, but not above that of the simpler classifier.

Table 1: Indicator features.

Name	Description
is-the	token x = "the"
is-free	token x = "free"
POS-tags	Part-of-speech tags of pair $x_{L0}\ x_{R0}$
fwd-pos	position from beginning, i
rev-pos	position from end $N - i$

Table 2: Statistical features.

Name	Description
web-count	count of "x" on the web
pair-count	web count "$w\ x$"
definite	web count "the $w\ x$"
collapsed	web count "wx" (one word)
and-count	web count "w and x"
genitive	web count "w's x"
Qcount-1	Counts of "x" in query database
Qcounts-2	Counts of "$w\ x$" in database

ken is 2, etc. The two lexical features (for when the token is "the" and when the token is "free") fire separately for the left and right tokens around the decision boundary. They are designed to add discrimination for these common query words, motivated by examples in our training set. For example, in the training set, "free" often occurs in its own segment when it's on the left-hand-side of a decision boundary (e.g. "free" "online" ...), but may join into a larger segment when it's on the right-hand-side of a collocation (e.g. "sulfite free" or "sugar free"). The classifier can use the feature weights to encourage or discourage segmentation in these specific situations.

For statistical features, previous work (Section 2) suggests that the mutual information between the decision tokens x_{L0} and x_{R0} may be appropriate. The log of the pointwise mutual information (Church and Hanks, 1989) between the decision-boundary tokens x_{L0}, x_{R0} is:

$$\text{MI}(x_{L0}, x_{R0}) = \log \frac{\text{Pr}(x_{L0}x_{R0})}{\text{Pr}(x_{L0})\text{Pr}(x_{R0})}$$

This is equivalent to the sum: $\log C(x_{L0}x_{R0}) + \log K - \log C(x_{L0}) - \log C(x_{R0})$. For web-based features, the counts $C(.)$ can be taken as a search engine's count of the number of pages containing the term. The normalizer K is thus the total number of pages on the Internet.

Represented as a summation, we can see that providing MI as the feature effectively ties the weights on the logarithmic counts $C(x_{L0}x_{R0})$, $C(x_{L0})$, and $C(x_{R0})$. Another approach would be to provide these logarithmic counts as separate features to our learning algorithm, which can then set the weights optimally for segmentation. We call this set of counts the "Basic" features. In Section 5, we confirm results on our development set that showed using the basic features untied increased segmentation performance by up to 4% over using MI – an important observation for all researchers using association models as features in their discriminative classifiers.

Furthermore, with this technique, we do not need to normalize the counts for the other pairwise statistical features given in Table 2. We can simply rely on our learning algorithm to increase or decrease the weights on the logarithm of the counts as needed.

To illustrate how the statistical features work, consider a query from our development set: "star wars weapons guns." The phrase "star wars" can easily be interpreted as a phrase; there is a high co-occurrence count (pair-count), and many pages where they occur as a single phrase (collapsed), e.g. "starwars.com." "Weapons" and "guns," on the other hand, should not be joined together. Although they may have a high co-occurrence count, the coordination feature (and-count) is high ("weapons and guns") showing these to be related concepts but not phrasal constituents. Including this novel feature resulted in noticeable gains on the development set.

Since this is a query-based segmentation, features that consider whether sets of tokens occurred elsewhere in the query database may provide domain-specific discrimination. For each of the Qcount features, we look for two quantities: the number of times the phrase occurs as a query on its own and the number of times the phrase occurs within another query.[4] Including both of these counts also resulted in performance gains on the development set.

We also extensively investigated other corpus-based features, such as the number of times the phrase occurred hyphenated or capitalized, and the

[4] We exclude counts from the training, development, and testing queries discussed in Section 4.1.

corpus-based distributional similarity (Lin, 1998) between a pair of tokens. These features are not available from search-engine statistics because search engines disregard punctuation and capitalization, and collecting page-count-based distributional similarity statistics is computationally infeasible.

Unfortunately, none of the corpus-based features improved performance on the development set and are thus excluded from further consideration. This is perhaps not surprising. For such a task that involves real user queries, with arbitrary spellings and sometimes exotic vocabulary, gathering counts from web search engines is the only way to procure reliable and broad-coverage statistics.

3.2.2 Context Features

Although the tokens at the decision boundary are of paramount importance, information from the neighbouring tokens is also critical for segmentation decision discrimination. We thus include features that take into consideration the preceding and following tokens, x_{L1} and x_{R1}, as context information. We gather all the token indicator features for each of these tokens, as well as all pairwise features between x_{L1} and x_{L0}, and then x_{R0} and x_{R1}. If context tokens are not available at this position in the query, a feature fires to indicate this. Also, if the context features are available, we include trigram web and query-database counts of "x_{L1} x_{L0} x_{R0}" and "x_{L0} x_{R0} x_{R1}", and a fourgram spanning both contexts. Furthermore, if tokens x_{L2} and x_{R2} are available, we collect relevant token-level, pairwise, trigram, and fourgram counts including these tokens as well.

In Section 5, we show that context features are very important. They allow our system to implicitly leverage surrounding segmentation decisions, which cannot be accessed directly in an independent segmentation-decision classifier. For example, consider the query "bank loan amoritization schedule." Although "loan amoritization" has a strong connection, we may nevertheless insert a break between them because "bank loan" and "amoritization schedule" each have even stronger association.

3.2.3 Dependency Features

Motivated by work in noun phrase parsing, it might be beneficial to check if, for example, token x_{L0} is more likely to modify a later token, such as x_{R1}. For example, in "female bus driver", we might not wish to segment "female bus" because "female" has a much stronger association with "driver" than with "bus". Thus, as features, we include the pairwise counts between x_{L0} and x_{R1}, and then x_{L1} and x_{R0}. Features from longer range dependencies did not improve performance on the development set.

4 Experimental Setup

4.1 Data

Our dataset was taken from the AOL search query database (Pass et al., 2006), a collection of 35 million queries submitted to the AOL search engine. Most punctuation has been removed from the queries.[5] Along with the query, each entry in the database contains an anonymous user ID and the domain of the URL the user clicked on, if they selected one of the returned pages. For our data, we used only those queries with a click-URL. This subset has a higher proportion of correctly-spelled queries, and facilitates annotation (described below).

We then tagged the search queries using a maximum entropy part-of-speech tagger (Ratnaparkhi, 1996). As our approach was designed particularly for noun phrase queries, we selected for our final experiments those AOL queries containing only determiners, adjectives, and nouns. We also only considered phrases of length four or greater, since queries of these lengths are most likely to benefit from a segmentation, but our approach works for queries of any length. Future experiments will investigate applying the current approach to phrasal verbs, prepositional idioms and segments with other parts of speech.

We randomly selected 500 queries for training, 500 for development, and 500 for final testing. These were all manually segmented by our annotators. Manual segmentation was done with improving search precision in mind. Annotators were asked to analyze each query and form an idea of what the user was searching for, taking into consideration the click-URL or performing their own online searches, if needed. The annotators were then asked to segment the query to improve search retrieval, by forcing a search engine to find pages with the segments

[5] Including, unfortunately, all quotation marks, precluding our use of users' own segmentations as additional labelled examples or feature data for our system

occurring as unbroken units.

One annotator segmented all three data sets, and these were used for all the experiments. Two additional annotators also segmented the final test set to allow inter-annotator agreement calculation. The pairwise agreement on segmentation decisions (between each pair of tokens) was between 84.0% and 84.6%. The agreement on entire queries was between 57.6% and 60.8%. All three agreed completely on 219 of the 500 queries, and we use this "intersected" set for a separate evaluation in our experiments.[6] If we take the proportion of segmentation decisions the annotators would be expected to agree on by chance to be 50%, the *Kappa* statistic (Jurafsky and Martin, 2000, page 315) is around .69, below the .8 considered to be good reliability.

This observed agreement was lower than we anticipated, and reflects both differences in query interpretation and in the perceived value of different segmentations for retrieval performance. Annotators agreed that terms like "real estate," "work force," "west palm beach," and "private investigator" should be separate segments. These are collocations in the linguistics sense (Manning and Schütze, 1999, pages 183-187); we cannot substitute related words for terms in these expressions nor apply syntactic transformations or paraphrases (e.g. we don't say "investigator of privates"). However, for a query such as "bank manager," should we exclude web pages that discuss "manager of the bank" or "branch manager for XYZ bank"? If a user is searching for a particular webpage, excluding such results could be harmful. However, for query substitution or expansion, identifying that "bank manager" is a single unit may be useful. We can resolve the conflicting objectives of our two motivating applications by moving to a multi-layer query bracketing scheme, first segmenting unbreakable collocations and then building them into semantic units with a query segmentation grammar. This will be the subject of future research.

4.2 Experiments

All of our statistical feature information was collected using the Google SOAP Search API.[7] For training and classifying our data, we use the popular Support Vector Machine (SVM) learning package SVM^{light} (Joachims, 1999). SVMs are maximum-margin classifiers that achieve good performance on a range of tasks. In each case, we learn a linear kernel on the training set segmentation decisions and tune the parameter that trades-off training error and margin on the development set.

We use the following two evaluation criteria:

1. Seg-Acc: *Segmentation decision accuracy*: the proportion of times our classifier's decision to insert a segment break or not between a pair of tokens agrees with the gold standard decision.

2. Qry-Acc: *Query segmentation accuracy*: the proportion of queries for which the complete segmentation derived from our classifications agrees with the gold standard segmentation.

5 Results

Table 3 provides our results for various configurations of features and token-combinations as described in Section 3.[8] For comparison, a baseline that always chooses a segmentation break achieves 44.8% Seg-Acc and 4.2% Qry-Acc, while a system that inserts no breaks achieves 55.2% Seg-Acc and 4.0% Qry-Acc. Our comparison system is the MI approach used by Jones et al. (2006), which achieves 68% Seg-Acc and 26.6% Qry-Acc (Table 3). We let the SVM set the threshold for MI on the training set.

Note that the *Basic*, *Decision-Boundary* system (Section 3.2.1), which uses exactly the same co-occurrence information as the MI system (in the form of the *Basic* features) but allows the SVM to discriminatively weight the logarithmic counts, immediately increases Seg-Acc performance by 3.7%. Even more strikingly, adding the *Basic* count information for the *Context* tokens (Section 3.2.2) boosts performance by another 8.5%, increasing Qry-Acc by over 22%. Smaller, further gains arise by adding *Dependency* token information (Section 3.2.3).

Also, notice that moving from *Basic* features for the *Decision-Boundary* tokens to all of our indicator (Table 1) and statistical (Table 2) features (referred to as *All* features) increases performance from 71.7% to 84.3%. These gains convincingly justify

[6] All queries and statistical feature information is available at http://www.cs.ualberta.ca/~bergsma/QuerySegmentation/

[7] http://code.google.com/apis/soapsearch/

[8] Statistically significant intra-row differences in Qry-Acc are marked with an asterix (McNemar's test, $p<0.05$)

Table 3: Segmentation Performance (%)

Feature Type	Feature Span	Test Set		Intersection Set	
		Seg-Acc	Qry-Acc	Seg-Acc	Qry-Acc
MI	Decision-Boundary	68.0	26.6	73.8	34.7
Basic	Decision-Boundary	71.7	29.2	77.6	39.7
Basic	Decision-Boundary, Context	80.2	52.0*	85.6	62.1*
Basic	Decision-Boundary, Context, Dependency	81.1	53.2	86.2	64.8
All	Decision-Boundary	84.3	57.8*	86.6	63.5
All	Decision-Boundary, Context	86.3	63.8*	89.2	71.7*
All	Decision-Boundary, Context, Dependency	85.8	61.0	88.7	69.4

our use of an expanded feature set for this task. Including *Context* with the expanded features adds another 2%, while adding *Dependency* information actually seems to hinder performance slightly, although gains were seen when adding *Dependency* information on the development set.

Note, however, that these results must also be considered in light of the low inter-annotator agreement (Section 4.1). Indeed, results are lower if we evaluate using the test-set labels from another annotator (necessarily training on the original annotator's labels). On the intersected set of the three annotators, however, results are better still: 88.7% Seg-Acc and 69.4% Qry-Acc on the intersected queries for the full-featured system (Table 3). Since high performance is dependent on consistent training and test labellings, it seems likely that developing more-explicit annotation instructions may allow further improvements in performance as within-set and between-set annotation agreement increases.

It would also be theoretically interesting, and of significant practical importance, to develop a learning approach that embraces the agreement of the annotations as part of the learning algorithm. Our initial ranking formulation (Section 3.1), for example, could learn a model that prefers segmentations with higher agreement, but still prefers any annotated segmentation to alternative, unobserved structures. As there is growing interest in making maximal use of annotation resources within discriminative learning techniques (Zaidan et al., 2007), developing a general empirical approach to learning from ambiguously-labelled examples would be both an important contribution to this trend and a potentially helpful technique in a number of NLP domains.

6 Conclusion

We have developed a novel approach to search query segmentation and evaluated this approach on actual user queries, reducing error by 56% over a recent comparison approach. Gains in performance were made possible by both leveraging recent progress in feature engineering for noun compound bracketing, as well as using a flexible, discriminative incorporation of association information, beyond the decision-boundary tokens. We have created and made available a set of manually-segmented user queries, and thus provided a new testing platform for other researchers in this area. Our initial formulation of query segmentation as a structured learning problem, and our leveraging of association statistics beyond the decision boundary, also provides powerful tools for noun compound bracketing researchers to both move beyond three-word compounds and to adopt discriminative feature weighting techniques.

The positive results achieved on this important application should encourage further inter-disciplinary collaboration between noun compound interpretation and information retrieval researchers. For example, analysing the semantics of multiword expressions may allow for more-focused query expansion; knowing to expand "bank manager" to include pages describing a "manager of the bank," but not doing the same for non-compositional phrases like "real estate" or "private investigator," requires exactly the kind of techniques being developed in the noun compound interpretation community. Thus for query expansion, as for query segmentation, work in natural language processing has the potential to make a real and immediate impact on search-engine technology.

The next step in this research is to directly investigate how query segmentation affects search performance. For such an evaluation, we would need to know, for each possible segmentation (including no segmentation), the document retrieval performance. This could be the proportion of returned documents that are deemed to be relevant to the original query. Exactly such an evaluation was recently used by Kumaran and Allan (2007) for the related task of query contraction. Of course, a dataset with queries and retrieval scores may serve for more than evaluation; it may provide the examples used by the learning module. That is, the parameters of the contraction or segmentation scoring function could be discriminatively set to optimize the retrieval of the training set queries. A unified framework for query contraction, segmentation, and expansion, all based on discriminatively optimizing retrieval performance, is a very appealing future research direction. In this framework, the size of the training sets would not be limited by human annotation resources, but by the number of queries for which retrieved-document relevance judgments are available. Generating more training examples would allow the use of more powerful, finer-grained lexical features for classification.

Acknowledgments

We gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, the Alberta Ingenuity Center for Machine Learning, and the Alberta Informatics Circle of Research Excellence.

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden markov support vector machines. In *ICML*.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. In *ACL*, pages 76–83.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(4):479–496.

Hal Daumé III and Daniel Marcu. 2004. NP bracketing by maximum entropy tagging and SVM reranking. In *EMNLP*, pages 254–261.

Thorsten Joachims. 1999. Making large-scale Support Vector Machine learning practical. In B. Schölkopf and C. Burges, editors, *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT-Press.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM Conference on Knowledge Discovery and Data Mining*, pages 133–142.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *WWW*, pages 387–396.

Daniel Jurafsky and James H. Martin. 2000. *Speech and language processing*. Prentice Hall.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Giridhar Kumaran and James Allan. 2007. A case for shorter queries, and helping users create them. In *NAACL-HLT*, pages 220–227.

Mirella Lapata and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks. In *HLT-NAACL*, pages 121–128.

Mark Lauer. 1995. Corpus statistics meet the noun compound: Some empirical results. In *ACL*, pages 47–54.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING/ACL*, pages 768–773.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Preslav Nakov and Marti Hearst. 2005a. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *CoNLL*, pages 17–24.

Preslav Nakov and Marti Hearst. 2005b. Using the web as an implicit training set: application to structural ambiguity resolution. In *HLT/EMNLP*, pages 835–842.

Jeremy Nicholson and Timothy Baldwin. 2006. Interpretation of compound nominalisations using corpus and web statistics. In *ACL Workshop on Multiword Expressions*, pages 54–61.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *The First International Conference on Scalable Information Systems*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*, pages 133–142.

Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. 2003. Query segmentation for web search. In *WWW (Poster Session)*.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *NAACL-HLT*, pages 260–267.

Chengxiang Zhai. 1997. Fast statistical parsing of noun phrases for document indexing. In *ANLP*, pages 312–319.

Bootstrapping Information Extraction from Field Books

Sander Canisius and **Caroline Sporleder**
ILK / Communication and Information Sciences
Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands
{S.V.M.Canisius,C.Sporleder}@uvt.nl

Abstract

We present two machine learning approaches to information extraction from semi-structured documents that can be used if no annotated training data are available, but there does exist a database filled with information derived from the type of documents to be processed. One approach employs standard supervised learning for information extraction by artificially constructing labelled training data from the contents of the database. The second approach combines unsupervised Hidden Markov modelling with language models. Empirical evaluation of both systems suggests that it is possible to bootstrap a field segmenter from a database alone. The combination of Hidden Markov and language modelling was found to perform best at this task.

1 Introduction

Over the past decades much textual data has become available in electronic form. Many text types are inherently more or less structured, for example, classified advertisements for appartments, medical records, or logs of archaeological finds or zoological specimens collected during expeditions. Such documents consist of a number of shorter texts (or *entries*), each describing an individual object (e.g., an appartment, or an archaeological find) or event (e.g., a patient presenting to a health care provider). These descriptions in turn typically consist of different segments (or *fields*) which contain information of a specific type drawn from a more or less given inventory. Example (1), for instance, shows two descriptions of zoological specimens (a snake and three frogs) collected during an expedition. The descriptions contain different segments giving information about the specimens and the circumstances of their collection. For example, in the first description, *Leptophis* and *ahaetulla* refer, respectively, to the genus and species of the specimen, *road to Overtoom* mentions the place of collection, *in bush above water* encodes information about the biotope, *in the process of eating Hyla minuta* is a remark about the circumstances of collection, *16-V-1968* gives the collection date and *RMNH 15100* the registration number.

(1) Leptophis ahaetulla, road to Overtoom, in bush above water in the process of eating Hyla minuta 16-V-1968. RMNH 15100

Hyla minuta 1 ♀ 2 ♂ Las Claritas, 9-VI-1978 quaking near water 50 cm above water surface, near secondary vegetation, 200 m, M.S. Hoogmoed, RMNH 27217 27219

Unfortunately, this inherent structure is rarely made explicit. While the different object or event descriptions might be indicated by additional whitespace or other formatting means, as in the example above, the individual fields within a description are typically not marked in any way. However, knowledge of the inherent structure would be very beneficial for information extraction and retrieval. For instance, texts in their raw form only allow key word search. To retrieve all entries describing specimens of type *Hyla minuta* from a zoological field report, one can only search for occurrences of that string anywhere in the document. This can return false

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 827–836, Prague, June 2007. ©2007 Association for Computational Linguistics

positives, such as the first description in (1) above, which does contain the string but is not about a *Hyla minuta* specimen but about a specimen of type *Leptophis ahaetulla* (the string *Hyla minuta* just happens to occur in the SPECIAL REMARKS field). On the other hand, if the genus and species information in an entry was explicitly marked, it would be possible to query specifically for entries whose GENUS is *Hyla* and whose SPECIES is *minuta*, thus avoiding the retrieval of entries in which this string occurs in another field.

The task of automatically finding and labelling segments in object or event descriptions has been referred to as *field segmentation* (Grenager et al., 2005).[1] It can be seen as a sequence labelling problem, where each text is viewed as a sequence of tokens and the aim is to assign each token a label indicating to what segment the token belongs (e.g., BIOTOPE or LOCATION). If training data in the form of texts annotated with segment information was readily available, the problem could be approached by training a sequence labeller in a supervised machine learning set-up. However, manually annotated data is rarely available. Creating it from scratch is not only time consuming but usually also requires a certain amount of expert knowledge. Moreover, the sequence labeller has to be re-trained for each new domain (e.g., natural history vs. archaeology) and possibly also each sub-domain (e.g., insects vs. mammals) due to the fact that the inventory of fields varies.

Thus, fully supervised machine learning is not feasible for this task. In this paper, we explore two approaches which require no or only a very small amount of manually labelled training data. Both approaches exploit the fact that there are often resources *derived* from the original documents that can potentially be utilised to bootstrap a sequence labeller in the absence of labelled training data. It is common practice, for example, that information contained in (semi-structured) field reports or medical records is manually entered into a database, usually in an attempt to make the data more accessible and easier to search. In such databases, each row corresponds to an entry in the original document (e.g., a zoological specimen) and the database columns correspond to the fields one would like to discern in the original document. Manually converting raw text documents into databases is a laborious task though, and it is rather common that the database covers only a small fraction of the objects described in the original texts. The research question we address in this paper is whether it is possible to bootstrap a domain-specific field segmentation system from an existing, manually created database for that domain. Such a system could then be applied to the remaining texts in that domain, which could then be segmented (semi-)automatically and possibly be added to the original database.

A database does not make perfect training material for a field segmenter though, as it is only derived from the original document and there are typically significant (and sometimes systematic) differences between the two data sources: First, while the ordering of the segments in a semi-structured text document is often not entirely fixed, some orderings are more likely than others. This information is lost in the derived databases. Second, the databases may contain information that is not normally present in the underlying text documents, for example information relating to the storage of an object in a collection. Conversely, some of the details present in the texts might be omitted from the database, e.g., the SPECIAL REMARKS field might be significantly shortened in the database. Third, pieces of information are frequently re-written when entered in the database, in some cases these differences may be systematic, e.g., dates, person names, or registration numbers might be written in a different format. Also, field boundaries in the text documents are sometimes indicated by punctuation, such as commas, and fields sometimes start with explicit key words, such as *collector*. Both of these features are missing from the database.

Despite of this, these databases will provide certain clues about the structure and content of different segments in the text documents. We exploit this in two different ways: (i) by concatenating database fields to artificially create annotated training data for a supervised machine learner, and (ii) by using the database to build language models for the field seg-

[1] The task differs from many other information extraction problems in which the aim is to extract short pieces of relevant information from larger text of largely irrelevant information. In *field segmentation*, all or most of the information in the input document is assumed to be relevant and the task is to segment it into fields containing different types of information.

mentation task.

2 Related Work

Most approaches to field segmentation and related information extraction tasks, such as filling templates with information about specific events, have been supervised. Freitag and Kushmerick (2000) combine a pattern learner with boosting to perform field segmentation in raw texts and in highly structured texts such as web pages and test this approach on a variety of field segmentation and template filling tasks. Kushmerick et al. (2001) address the problem of extracting contact information from business cards. They mainly focus on field *labelling*, bypassing the segmentation step by assuming that each line on a business card only contains one field (though a field like ADDRESS may span several lines). Their method combines a text classifier, for assigning likely labels to each field, with a trained Hidden Markov Model (HMM) for learning ordering constraints between fields. Borkar et al. (2001) identify fields in international postal addresses and bibliographic records by nesting HMMs: an outer HMM for modelling field transitions and a number of inner HMMs for modelling token transitions within fields. Viola and Narasimhand (2005) also deal with address segmentation but employ a trained context-free grammar.

One of the few unsupervised approaches is provided by Grenager et al. (2005), who perform field segmentation on bibliographic records and classified advertisements, using EM to fit an HMM to the data. They show that an unconstrained model does not learn the field structure very well and propose augmenting the model with a limited amount of domain-unspecific background knowledge, for example, by modifying the transition model to bias it towards recognising larger-scale patterns.

3 Learning Field Segmentation from Databases

3.1 Data

We tested our approach on two datasets provided by Naturalis, the Dutch National Museum of Natural History[2] Each dataset consists of (i) a number of field book entries describing the circumstances under which animal specimens were collected, and (ii) a database containing similar information about the same group of animals but in a more structured form. The latter were used for training, the former for testing. While the databases were manually created from the corresponding field books, we made sure that the field book entries we selected for testing did not overlap with the database entries. The two data sets are described below. Table 1 lists the main properties of the data.

Reptiles and Amphibians (RA) This dataset describes a number of reptile and amphibian specimens. The database consists of 16,670 entries and 41 columns. The columns relate, for example, to the circumstances of a specimen's collection, its taxonomic classification, how and where it is stored, who entered the entry into the database and when. Many database cells are empty. Those that are filled come in a variety of format, i.e., numbers, dates, individual words, and free text of various lengths. 22 of the columns contained information that was missing from the field books, e.g., information relating to the storage of the specimens; these columns were excluded from the experiments.

From the corresponding field books, 210 entries were selected randomly and manually annotated with segment information. To test the reliability of the manual annotation, 50 entries were labelled by two annotators. The inter-annotator accuracy on the token level was 92.84% and the kappa .92. The number of distinct field types found in the entries was 19, some of which only occurred in two entries, others occurred in virtually every entry. The average field length was four tokens, with a maximum average of 21 for the SPECIAL REMARKS field, and a minimum of one for fields such as SPECIES. The average number of tokens per entry was 60. Punctuation marks that did not clearly belong to any field were labelled as OTHER. In the experiments, 200 entries were used for testing and 10 for parameter tuning.

Pisces The second dataset contains information about the stations where fish specimens were caught. The database consists of 1,375 entries and four columns which provide information on the location of the stations. From the corresponding field books, we manually labelled 100 entries. Compared to the

[2] http://www.naturalis.nl

	RA	Pisces
# entries in DB	16,670	1,375
# fields	19	4
entry length (avg.)	60.17	39.79
segment length (avg.)	4.08	4.75

Table 1: Properties of the two datasets

first data set, this set is much more regular, with less variation in the number of segments per entry and in the average segment length. The field book entries are also much shorter and there are fewer segments (see Table 1).

3.2 Baselines

In order to get a sense of the difficulty of the task, we implemented five baseline approaches. For the first, **Majority** (**MajB**), we always assign the field label that occurs most frequently in the manually labelled test data, namely SPECIAL REMARKS. The other four baselines implement different look-up strategies, using the database to determine which label should be assigned to a token or token sequence.

Exact (**ExactB**) looks for substrings in a field book entry which exactly match the content of a database cell and then assigns each token in the matched string the corresponding column label from the database. There are normally several ways to match a field book entry to the database cells; we employed a greedy search, labelling the longest matching substrings first. All tokens that could not be matched in this way were assigned the label OTHER.

Unigram (**UniB**) assigns each token the column label of the database cell in which it occurs most frequently. If a token is not found in the database, it is assigned the label OTHER.

Trigram (**TriB**) assigns each token the most frequent column label of the trigram centred on it. If a trigram is not found in the database, the baseline backs off to the two bigrams covering the token and then to the unigram. If the token is not found in the database, OTHER is assigned.

Trigram+Voting (**TriB+Vote**) is based on a technique proposed by Van den Bosch and Daelemans (2005) for sequence labelling tasks. The main idea is to assign labels to trigrams in the sequence using a sliding window. Because each token, except the boundary tokens, is contained in three different trigrams (i.e., the one centred on the token to its left, the one centred on itself, and the one centred on the token to its right), each token gets three labels assigned to it, over which voting can be performed. In our case the labels are assigned by database look-up. If a trigram is not found in the database, no label is assigned to it. If the labels assigned to a given token differ, majority voting is used to resolve the conflict. If this does not break the tie (i.e., because all three trigrams assign different labels), the label of the trigram that occurs most frequently in the database is assigned. We also implemented two post-processing rules: (i) turning the label OTHER between two identical neighbouring labels into the surrounding labels, and (ii) labelling commas as OTHER if the neighbouring labels are not identical.

3.3 Supervised Learning from Automatically Generated Training Data

Our first strategy was to automatically generate training data for a supervised machine learner from the database. Since the rows in the database correspond to field book entries and the columns corresponds to the fields that we want to identify, training data can be obtained by concatenating the cells in each database row. The order of the fields in the field book entries is not fixed and this should also be reflected in the artificially generated training data. However, the field sequence is not entirely random, i.e., not all sequences are equally likely. If a small amount of manually annotated data is available, the field transition probabilities can be estimated from this, otherwise the best one can do is to assume uniform probabilities for all possible orderings. We experimented with both strategies, creating two different training sets, one in which the database cells were concatenated randomly with uniform probabilities, and another in which the cells were concatenated to reflect the field ordering probabilities estimated from ten entries in the manually labelled development set.[3] When estimating the field transition

[3] We found that 10 annotated entries are enough for this purpose; the field segmentation results we obtained by estimating the sequence probabilities for the training set from 100 entries were not significantly different. This is probably because the probabilities are only used indirectly, i.e. to bias the field orderings for the generated training data. If the probabilities were used directly in the model, the amount of manually annotated data would probably matter much more.

probabilities, we computed a probability distribution over the initial fields of an entry as well as the conditional probability distributions of a field x following a field y for all seen segment pairs in the ten entries. To account for unobserved events, we used Laplace smoothing.

The artificially created training data were then converted to a token-based representation in which each token corresponds to an instance to be labelled with the field to which it belongs. On the whole, we had just under 700,000 instances (i.e., tokens) in our training data. We implemented 107 features, falling in three classes:

- the neighbouring tokens (in a window of 5 centering on the token in focus)
- the typographic properties of the focus token (word vs. number, capitalisation, number of characters in the token etc.)
- the tfidf weight of the focus token in its context with respect to each of the columns in the database (i.e., the fields)

The tfidf-based features were computed for a window of three, centering on the token in focus. For all n-grams in this window covering the token in focus (i.e., the trigram, the two bigrams, and the unigram of the focus token), we calculated the $tfidf$ similarity with the columns in the database, where the similarity between an n-gram t_i and a column col_x is defined as:

$$tfidf_{t_i,col_x} = tf_{t_i,col_x} \log idf_{t_i}$$

The term frequency, tf_{t_i,col_x} is the number of occurrences of t_i in col_x divided by the number of occurrences of all n-grams of length n in col_x (0 if the n-gram does not occur in the column). The inverse document frequency, idf_{t_i}, is the number of all columns in the database divided by the number of columns containing t_i. A high tfidf weight for a given n-gram in a given column means that it frequently occurs in that column but rarely in other columns, thus it is a good indicator for that column.

The training data was then used to train a memory-based machine learner (TiMBL (Daelemans et al., 2004), default settings, $k = 3$, numeric features declared) to determine which field each token belongs to.[4]

3.4 Hidden Markov Models

Our second approach combines language modelling and Hidden Markov Models (HMMs) (Rabiner, 1989). Hidden Markov Models have been in use for information extraction tasks for a long time. A probabilistic model is trained to assign a label, or *state* to each of a sequence of observations, where both labels and observations are expected to be sequentially correlated; hence the popularity of HMMs in natural language processing and information extraction. Recently, a large number of more sophisticated learning techniques have largely replaced HMMs for information extraction; however unlike most of those newer techniques, HMMs offer the advantage of having a well-established unsupervised training procedure: the Baum-Welch algorithm (Baum et al., 1970).

Training a Hidden Markov Model, whether supervised or unsupervised, comes down to estimating three probability distributions.

1. An initial state distribution π, which models the probability of the first observation of a sequence to have a certain label.

2. A state-transition distribution A, modelling the conditional probability of being in a certain state s, given that the previous state was s'.

3. A state-emission distribution B, which models the conditional probability of observing a certain object o given some state s.

For information extraction tasks, the typical interpretation of an *observation* as referred to above, is that of a token, where the entire observation sequence commonly corresponds to one sentence. In the current study, we chose to apply HMMs on a somewhat higher level, where an observation corresponds to a *segment* of the field book entry. Ideally, one such segment maps one-to-one to a cell in the specimen database, though we leave open the possibility of merging several segments into one database cell.

Provided that a field book entry can be segmented reliably, we have turned one part of the learning problem, that of estimating the state-emission

[4] We chose TiMBL because it has been applied successfully to sequence labelling tasks (Van den Bosch and Canisius, 2006; Van den Bosch and Daelemans, 2005).

distribution, into one for which we have (almost) perfect supervised training data: the contents of the database cells. The general form of a Hidden Markov Model's state-emission distribution is $P(o|s)$, where s is the state, i.e. a field type in our case, and o is the observation. As mentioned before, we treat a segment of tokens as one observation, therefore our state-emission distribution will look like $P(o = t_1, t_2, ..., t_n|s)$. Essentially, what we have here is a language model, conditioned on the current state. Since the specimen database provides a large amount of labelled segment sequences, any probabilistic language modelling method can be used to estimate the state-emission distribution.

Whereas the specimen database provides sufficient information to estimate the state-emission distribution in a fully supervised way, the initial-state and state-transition distributions cannot be derived from the database alone. Columns in a database are either unordered or ordered in a way that does not necessarily reflect the order they had in the field book entries they were extracted from. However, the original field book entries do show a rather systematic structure. Often, using information about the order fields typically occur in, seems to be the only way to distinguish certain field types from one another. To estimate the two missing probability distributions, the Baum-Welch algorithm was used, updating the initial-state and state-transition distributions, while keeping the state-emission distributions unchanged.

3.4.1 Segmentation of Field Book Entries

In our setup, the Hidden Markov Model expects the input texts to be pre-segmented. To come up with a good initial segmentation of an input entry, we again chose a language-modelling approach. It is expected that segment boundaries can best be recognised by looking for unusual token subsequences; that is, token sequences that are highly unlikely to occur within a field according to the information we obtained from the specimen database about what a typical segment does look like. A bigram language model has been trained on the contents of *all* the columns of the specimen database. Using this language model and the Viterbi algorithm, the globally most-likely segmentation of the input text is predicted.

3.4.2 The State-emission Model

The state-emission model is constructed by training a separate bigram language model for each column of the specimen database. Combining those gives us the conditional distribution required for a Hidden Markov Model. However, in a database, not every column has necessarily been filled for every record. For example, in the Reptiles and Amphibians database, there are columns that only contain actual data as infrequently as in 5% of the records. Relative to columns that contain data more often, these sparsely-filled columns tend to be overestimated when simply computing a likelihood according to the language model. For this reason, a penalty term is added to the state-emission distribution corresponding to the probability that a record contains data for the given column. The likelihood computed by the language model and the corresponding penalty term are then simply multiplied.

3.4.3 Language Modelling

For building both types of language model presented in the two previous sections, we used n-gram language modelling as implemented by the SRI Language Modelling Toolkit (Stolcke, 2002). With this toolkit, high-order n-gram models can be built, where the sparsity problem often encountered with such models is tackled by various smoothing methods. We supplemented this built-in n-gram smoothing, with our own smoothing on the token level by replacing low-frequent words with symbols reflecting certain orthographic features of the original word, and numbers with a symbol only encoding the number of digits in the original number.

In addition to these general measures to deal with sparsity, we also applied a small number of knowledge-driven modifications to the training data for the language models. The need for those is caused by the fact that the contents of the specimen database are almost, but not entirely extracted literally from the original field book entries. For example, for the second entry of Example 1, the following information is stored in the database.

Genus Hyla

Species minuta

Gender 1 f + 2 m

Place Las Claritas

Collection date 9-6-1978

Biotope quaking near water 50 cm above water surface, near secondary vegetation

Height 200 m

Collector M.S. Hoogmoed

Registration number 27217 27219

Comparing just this single field book entry with its corresponding database record, one can already see several mismatches. The gender symbols ♂ and ♀ in the field book texts are stored as `m` and `f` in the database. The collection date 9-VI-1978, has been converted to 9-6-1978 before adding it to the database, i.e. the Roman numeral for the month number has been mapped to the corresponding Arabic numeral. As a final example, in the field book entry the registration number for the specimen is preceded by the symbol `RMNH`; in the database this standard symbol is stripped of and only the number is stored. Each of these differences, while only small, will hinder the performance of a language model trained on the contents of the database and applied to field book texts.

As a simple illustration of this, when encountering the symbol `RMNH` in a field book entry, this most likely indicates the start of a new (registration number) segment. However, in the database, on which all language models are trained, `RMNH` never occurs as a symbol in the registration number column; it does occur a few times in the column for special remarks but never at the start of the text. As a result, a segmentation model trained on the contents of the database, on encountering the symbol `RMNH` will always opt for continuing the existing segment as opposed to starting a new one, which is most likely the better choice.

Fortunately, many such mismatches between the text in field books and the database are systematic and can easily be covered by a small number of manually constructed rules that modify the training data for the language models. Among others, we added the `RMNH` symbol in front of registration numbers, and randomly changed some month numbers from Arabic numerals to Roman numerals.

Another difference between the field books and the database that turned out to be rather crucial is the fact that many segments in the field book entries are separated by commas. Such commas used as delimiters *between fields* do not appear in the database, where fields correspond to columns and boundaries between fields consequently do not have to be explicitly marked by punctuation. For example, the comma between Las Claritas and 9-VI-1978 only serves to separate the *Place* segment from the *Collection date* segment; the comma is not copied to the database. However, commas do occur *field-internally* in the database, especially in longer fields such as SPECIAL REMARKS. Hence a language model trained on the database in its original form will never have encountered a comma functioning as a segment boundary marker and thus will not recognise that commas may be used for this purpose in the field book entries. To deal with this, we modified the training data for the segment model by randomly inserting commas at the end of some segments. Experimental results point out that this modification has a large impact on the performance of the segmentation model.

	Token	Segment			
	Acc.	Prec.	Rec.	$F_{\beta=1}$	WDiff
MajB	24.8	0.0	0.0	0.0	.346
ExactB	16.0	25.7	23.1	24.3	.425
UniB	27.0	8.9	22.8	12.8	.818
TriB	43.8	12.9	24.8	16.9	.582
TriB+Vote	45.1	14.9	27.8	19.4	.536
MBL rand.	44.6	7.1	19.2	10.4	.568
MBL bias	53.4	12.1	32.0	17.6	.533
HMM	56.9	62.7	58.1	60.3	.177

Table 2: Performance of all baseline and learning approaches on the Reptiles and Amphibians data, expressed in token accuracy, precision, recall, F-score, and WindowDiff. For WindowDiff, lower scores are better.

3.5 Results and Discussion

To evaluate the performance of the two approaches, we applied them to the Reptiles and Amphibians database. First we computed baseline scores using the approaches described in Section 3.2. All resulting scores are listed in Table 2.

Performance of the systems was measured using a number of different metrics, each reflecting different qualities of the output. The most basic one, token accuracy, simply measures the percentage of tokens that were assigned the correct field type. It has the disadvantage that it does not reflect the quality of the segments that were found. For a more segment-oriented evaluation, we used precision, recall and F-score on correctly identified and labelled segments. As a last measure for segmentation quality we used WindowDiff (Pevzner and Hearst, 2002), which only evaluates segment boundaries not the labels assigned to them. In comparison with F-score, it is more forgiving with respect to segment boundaries that are slightly off.

The baseline performance scores support our assumption that the contents of the database can be used to learn how to segment and label field book entries, i.e. the increasingly more sophisticated database matching strategies each cause a substantial performance improvement up to 45.1 token accuracy for the trigram lookup with voting strategy. The biggest problem of all baseline approaches is that their performance with respect to the segment-oriented measures is disappointing. Even trigram lookup with voting only reaches an F-score of 19.4.

Looking at the performance of the two memory-based learners in Table 2 (*MBL rand.* was trained on randomly concatenated training data, *MBL bias* on data modelled after 10 training sequences), we see that the small amount of prior knowledge used for generating the artificial training data results in a substantial improvement compared with the memory-based learner that was trained on randomly concatenated training data with uniform probabilities.

As can be seen in the last row of Table 2, the Hidden Markov Model outperforms all other approaches in all aspects; it attains both the best token accuracy (56.9), and by far the best F-score (60.3). The most probable explanation for the superior performance of the HMM-based approach is that this approach models sequential constraints between different segments, whereas the baselines and the memory-based models are predominantly local.

In Table 3, we consider the effect that the knowledge-based rewriting rules discussed in Subsection 3.4.3 have on the performance of both the segmentation and the labelling step. We evaluate both (i) the performance of the two processing steps separately—for labelling this presupposes perfectly segmented input— and (ii) the performance of the cascade of segmentation and labelling. As before, the performance of the labelling and the cascade is expressed in F-score on segments. Performance of the segmentation is measured in F-score on inserted segment *boundaries*.

The first row of the table shows the scores if no modification rules are used. This proves detrimental for the segmentation, only attaining an F-score of 28.4. With 62.3, the F-score for labelling is reasonable; however, the weakness of the segmentation causes the output of the entire cascade to be useless. Modifying the training data for the segmentation model by randomly inserting a comma at the end of segments gives a substantial improvement in segmentation performance, and as a result the quality of the cascade improves with it, as can be seen in the second row. All remaining rows list the scores of a single modification rule applied to the training data *in addition to* the comma rule. Each of the rules gives a slight performance increase. Using all rules together makes a big difference: the F-score of the cascade increases from 44.7 with the comma rule only, to 60.3.

Rule	Boundaries	Labels	Cascade
None	28.4	62.3	17.2
Comma	69.7	62.3	44.7
Comma+Collection Date	69.7	64.4	46.8
Comma+Reg. Number	72.9	68.6	50.9
Comma+Gender	71.5	65.0	49.5
Comma+Collector	70.4	65.8	45.3
All	72.0	78.3	60.3

Table 3: The effect of systematically modifying the training data for both the segmentation and labelling models. The comma rule is used only in the training of the segmentation model. The other rules are named after the database field they are applied to. The scores reflect their performance when applied in conjunction with the comma rule.

To confirm that the HMM-based approach carries over to other datasets, we also tested it on the Pisces data. The results of this experiment, as well as all baseline scores are presented in Table 4.[5] The fact that this data set is more regular and contains fewer segments is reflected by the relatively high token accuracies attained by the baseline approaches. With

[5] We did not test the memory-based approaches as these led to significantly worse results than the HMM-based model on the Reptiles and Amphibians data set.

the simple database lookup strategies, however, almost no entire segments are predicted correctly. Attaining a token accuracy of 94.4 and an F-score of 86.9, the performance of the Hidden Markov Model is again more than satisfactory, confirming the results observed with the Reptiles and Amphibians data.

	Token		Segment		
	Acc.	Prec.	Rec.	$F_{\beta=1}$	WDiff
MajB	50.0	0.0	0.0	0.0	.279
ExactB	9.3	0.0	0.0	0.0	.565
UniB	53.7	1.5	3.6	2.1	.588
TriB	68.6	0.2	0.2	0.2	.359
TriB+Vote	67.1	2.2	2.8	2.4	.384
HMM	94.4	87.6	86.3	86.9	.049

Table 4: Performance of Hidden Markov Model and all baseline approaches on the Pisces data, expressed in token accuracy, precision, recall, F-score, and WindowDiff. For WindowDiff, lower scores are better.

4 Conclusion

Information extraction is often used to automate the process of filling a structured database with content extracted from written texts. Supervised machine learning approaches have been successfully applied for creating systems capable of performing this task. However, the supervised nature of these approaches requires large amounts of annotated training data; the acquisition of which is often a laborious and time-consuming process. In this study, we experimented with two machine learning techniques that do not require such annotated training data, but can be trained on a database containing information derived from the type of documents targeted by the application.

The first approach is an attempt to employ a standard supervised machine learning algorithm, training it on artificial labelled training data. These data are created by concatenating the contents of the cells of the database records in random order. Experiments with this approach pointed out that truly random concatenation of database fields results in weak performance; a rather simple baseline approach, which only matches substrings of a field book entry with the contents of the database, leads to better results. However, if a small amount of annotated field book entries is available —in this study, 10 entries turned out to be sufficient— one can estimate field ordering probabilities that can be used to generate more realistic training data from the database. A machine learner trained on these data labelled 10% more tokens correctly than the system trained on the randomly generated data.

Our second approach is based on unsupervised Hidden Markov modelling. First, an n-gram language model is used to divide the field book entries into unlabelled segments. Then, a Hidden Markov Model is trained on these segmented entries using the Baum-Welch algorithm to estimate state-transition probabilities. The resulting HMM labels the segments found in the preceding segmentation step. The HMM state-emission distributions are estimated by training n-gram language models on the contents of the database columns.

The performance of the HMM proved to be superior to the other approaches, outperforming the supervised learner by labelling 56.9% of the tokens correctly, as well as attaining good results in terms of segment-level F-score (60.3). Experiments with the HMM approach on a second, independent data set confirmed its generality.

Acknowledgements

The research reported in this paper was funded by the Netherlands Organisation for Scientific Research (NWO) as part of the IMIX and CATCH programmes. We are grateful to Antal van den Bosch, Marieke van Erp, Steve Hunt, and the staff at Naturalis, the Dutch National Museum for Natural History, for interesting discussions and help in preparing the data.

References

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Vinayak Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. 2001. Automatic segmentation of text into structured records. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 175–186.

Walter Daelemans, Jakub Zavrel, Ko Van der Sloot, and Antal Van den Bosch, 2004. *TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide*. ILK Research Group Technical Report Series no. 04-02.

Dayne Freitag and Nicholas Kushmerick. 2000. Boosted wrapper induction. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI/IAAI-2000)*, pages 577–583.

Trond Grenager, Dan Klein, and Christopher D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 371–378.

Nicholas Kushmerick, Edward Johnston, and Stephen McGuinness. 2001. Information extraction by text classification. In *Proceedings of the IJCAI-01 Workshop on Adaptive Text Extraction and Mining*.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Lawrence R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. *Proceedings of the International Conference on Spoken Language Processing (ICSLP 2002)*, 2:901–904.

Antal Van den Bosch and Sander Canisius. 2006. Improved morpho-phonological sequence processing with constraint satisfaction inference. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON '06)*.

Antal Van den Bosch and Walter Daelemans. 2005. Improving sequence segmentation learning by predicting trigrams. In *Proceedings of the Ninth Conference on Natural Language Learning, CoNLL-2005*, pages 80–87.

Paul Viola and Mukund Narasimhand. 2005. Learning to extract information from semi-structured text using a discriminative context free grammar. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 330–337.

Extracting Data Records from Unstructured Biomedical Full Text

Donghui Feng **Gully Burns** **Eduard Hovy**
Information Sciences Institute
University of Southern California
Marina del Rey, CA, 90292
{donghui, burns, hovy}@isi.edu

Abstract

In this paper, we address the problem of extracting data records and their attributes from unstructured biomedical full text. There has been little effort reported on this in the research community. We argue that semantics is important for record extraction or finer-grained language processing tasks. We derive a data record template including semantic language models from unstructured text and represent them with a discourse level Conditional Random Fields (CRF) model. We evaluate the approach from the perspective of Information Extraction and achieve significant improvements on system performance compared with other baseline systems.

1 Introduction

The discovery and extraction of specific types of information, and its (re)structuring and storage into databases, are critical tasks for data mining, knowledge acquisition, and information integration from large corpora or heterogeneous resources (e.g., Muslea et al., 2001; Arasu and Garcia-Molina, 2003). For example, webpages of products on Amazon may contain a list of data records such as books, watches, and electronics. Automatic extraction of individual records will facilitate the access and management of data resources.

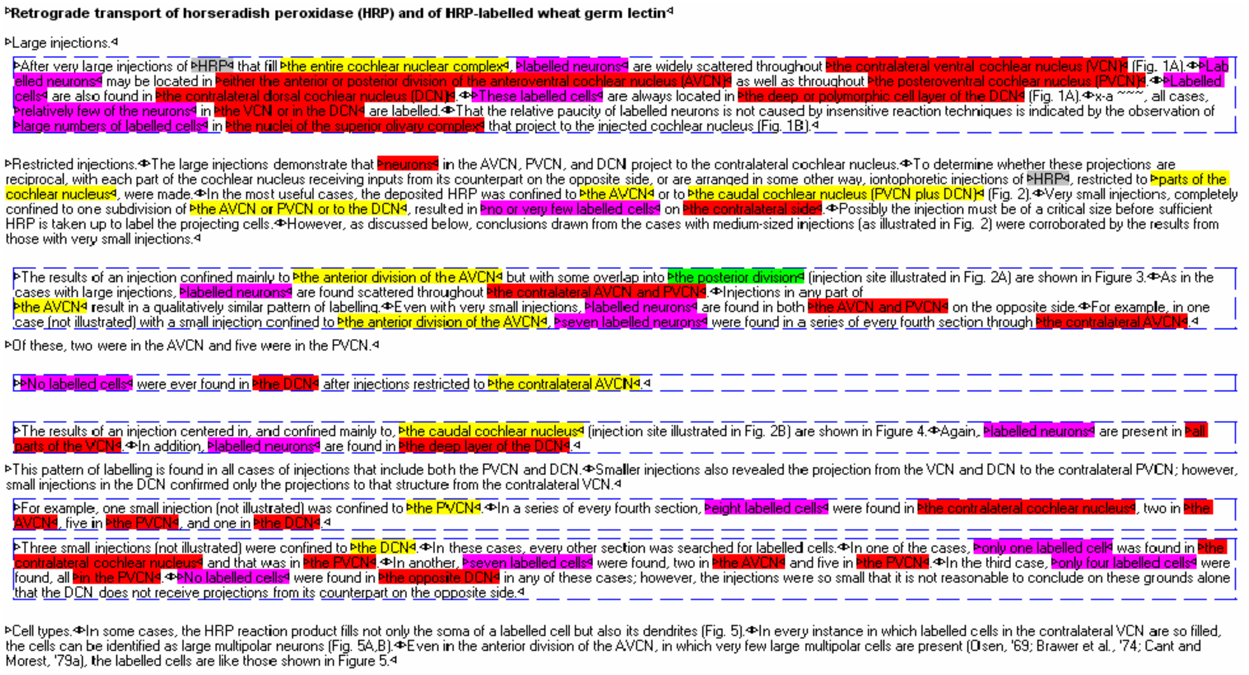
Most current approaches address this problem for structured or semi-structured text, for instance, from XML format files or lists and/or tabular data records on webpages (e.g., Liu et al., 2003; Zhu et al., 2006). The techniques applied rely strongly on the analysis of document structure derived from the webpage's html tags (e.g., the DOM tree model).

Regarding unstructured text, most Information Extraction (IE) work has focused on named entities (people, organizations, places, etc.). Such IE treats each extracted element as a separate record. Much less work has focused on the case where several related pieces of information have to be extracted to jointly comprise a single data record. In this work, it is usually assumed that there is only one record for each document (e.g., Kristjannson et al., 2004). Almost no work tries to extract multiple data records from a single document. Multiple data records can be scattered across the narrative in free text. The problem becomes much harder as there are no explicit boundaries between data records and no heavily indicative format features (like html tags) to utilize.

With the exponential increase of unstructured text resources (e.g., digitalized publications, papers and/or technical reports), knowledge needs have made it a necessity to explore this problem. For example, biomedical papers contain numerous experiments and findings. But the large volume and rate of publication have made it infeasible to read through the articles and manually identify data records and attributes.

We present a study to extract data records and attributes from the biomedical research literature. This is part of an effort to develop a Knowledge Base Management System to benefit neuroscience research. Specifically we are interested in knowledge of various aspects (attributes) of Tract-tracing Experiments (TTE) (data records) in neuroscience. The goal of TTE experiments is to chart the interconnectivity of the brain by injecting tracer chemicals into a region of the brain and identifying corresponding labeled regions where the tracer is

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 837–846, Prague, June 2007. ©2007 Association for Computational Linguistics



▹Retrograde transport of horseradish peroxidase (HRP) and of HRP-labelled wheat germ lectin◃

▹Large injections.◃

▹After very large injections of ▹HRP◃ that fill ▹the entire cochlear nuclear complex◃, ▹labelled neurons◃ are widely scattered throughout ▹the contralateral ventral cochlear nucleus (VCN)◃ (Fig. 1A).◃▹▹Labelled neurons◃ may be located in ▹either the anterior or posterior division of the anteroventral cochlear nucleus (AVCN)◃ as well as throughout ▹the posteroventral cochlear nucleus (PVCN)◃.◃▹▹Labelled cells◃ are also found in ▹the contralateral dorsal cochlear nucleus (DCN)◃.◃▹▹These labelled cells◃ are always located in ▹the deep or polymorphic cell layer of the DCN◃ (Fig. 1A).◃▹x-a ~~~, all cases, ▹relatively few of the neurons◃ in ▹the VCN or in the DCN◃ are labelled.◃▹That the relative paucity of labelled neurons is not caused by insensitive reaction techniques is indicated by the observation of ▹large numbers of labelled cells◃ in ▹the nuclei of the superior olivary complex◃ that project to the injected cochlear nucleus (Fig. 1B).◃

▹Restricted injections.◃▹The large injections demonstrate that ▹neurons◃ in the AVCN, PVCN, and DCN project to the contralateral cochlear nucleus.◃▹To determine whether these projections are reciprocal, with each part of the cochlear nucleus receiving inputs from its counterpart on the opposite side, or are arranged in some other way, iontophoretic injections of ▹HRP◃, restricted to ▹parts of the cochlear nucleus◃, were made.◃▹In the most useful cases, the deposited HRP was confined to ▹the AVCN◃ or to ▹the caudal cochlear nucleus (PVCN plus DCN)◃ (Fig. 2).◃▹Very small injections, completely confined to one subdivision of ▹the AVCN or PVCN or to the DCN◃, resulted in ▹no or very few labelled cells◃ on ▹the contralateral side◃.◃▹Possibly the injection must be of a critical size before sufficient HRP is taken up to label the projecting cells.◃▹However, as discussed below, conclusions drawn from the cases with medium-sized injections (as illustrated in Fig. 2) were corroborated by the results from those with very small injections.◃

▹The results of an injection confined mainly to ▹the anterior division of the AVCN◃ but with some overlap into ▹the posterior division◃ (injection site illustrated in Fig. 2A) are shown in Figure 3.◃▹As in the cases with large injections, ▹labelled neurons◃ are found scattered throughout ▹the contralateral AVCN and PVCN◃.◃▹Injections in any part of ▹the AVCN◃ result in a qualitatively similar pattern of labelling.◃▹Even with very small injections, ▹labelled neurons◃ are found in both ▹the AVCN and PVCN◃ on the opposite side.◃▹For example, in one case (not illustrated) with a small injection confined to ▹the anterior division of the AVCN◃, ▹seven labelled neurons◃ were found in a series of every fourth section through ▹the contralateral AVCN◃.◃

▹Of these, two were in the AVCN and five were in the PVCN.◃

▹▹No labelled cells◃ were ever found in ▹the DCN◃ after injections restricted to ▹the contralateral AVCN◃.◃

▹The results of an injection centered in, and confined mainly to, ▹the caudal cochlear nucleus◃ (injection site illustrated in Fig. 2B) are shown in Figure 4.◃▹Again, ▹labelled neurons◃ are present in ▹all parts of the VCN◃.◃▹In addition, ▹labelled neurons◃ are found in ▹the deep layer of the DCN◃.◃

▹This pattern of labelling is found in all cases of injections that include both the PVCN and DCN.◃▹Smaller injections also revealed the projection from the VCN and DCN to the contralateral PVCN; however, small injections in the DCN confirmed only the projections to that structure from the contralateral VCN.◃

▹For example, one small injection (not illustrated) was confined to ▹the PVCN◃.◃▹In a series of every fourth section, ▹eight labelled cells◃ were found in ▹the contralateral cochlear nucleus◃, two in ▹the AVCN◃, five in ▹the PVCN◃, and one in ▹the DCN◃.◃

▹Three small injections (not illustrated) were confined to ▹the DCN◃.◃▹In these cases, every other section was searched for labelled cells.◃▹In one of the cases, ▹only one labelled cell◃ was found in ▹the contralateral cochlear nucleus◃ and that was in ▹the PVCN◃.◃▹In another, ▹seven labelled cells◃ were found, two in ▹the AVCN◃ and five in ▹the PVCN◃.◃▹In the third case, ▹only four labelled cells◃ were found, all ▹in the PVCN◃.◃▹▹No labelled cells◃ were found in ▹the opposite DCN◃ in any of these cases; however, the injections were so small that it is not reasonable to conclude on these grounds alone that the DCN does not receive projections from its counterpart on the opposite side.◃

▹Cell types.◃▹In some cases, the HRP reaction product fills not only the soma of a labelled cell but also its dendrites (Fig. 5).◃▹In every instance in which labelled cells in the contralateral VCN are so filled, the cells can be identified as large multipolar neurons (Fig. 5A,B).◃▹Even in the anterior division of the AVCN, in which very few large multipolar cells are present (Osen, '69; Brawer et al., '74; Cant and Morest, '79a), the labelled cells are like those shown in Figure 5.◃

Figure 1. An example of data records and attributes in a research article.

taken up and transported to (Burns et al., 2007).

To extract data records from the research literature, we need to solve two sub-problems: discovering individual attributes of records and grouping them into one or more individual records, each record representing one TTE experiment. Each attribute may contain a list of words or phrases and each record may contain a list of attributes.

Listing each sentence from top to bottom, we call the first problem the *Horizontal Problem* (HP) and the second the *Vertical Problem* (VP). Figure 1 provides an example of a TTE research article with colored fragments representing attributes and dashed frames representing data records. For instance, the third dashed frame represents one experiment record having three attributes with corresponding biological interpretations: “no labeled cells”, “the DCN”, and “the contralateral AVCN”.

We view the HP and VP problems as two sequential labeling problems and describe our approach using two-level Conditional Random Fields (CRF) (Lafferty et al., 2001) models to extract data records and their attributes.

The HP problem (finding individual attribute values) is solved using a sentence-level CRF labeling model that integrates a rich set of linguistic features. For the VP problem, we apply a discourse-level CRF model to identify individual experiments (data records). This model utilizes deep semantic knowledge from the HP results (attribute labels within sentences) together with semantic language models and achieves significant improvements over baseline systems.

This paper mainly focuses on the VP problem, since linguistic features for the HP problem is the general IE topic of much past research (e.g., Peng and McCallum, 2004). We apply various feature combinations to learn the most suitable and indicative linguistic features.

The remainder of this paper is organized as follows: in the next section we discuss related work. Following that, we present the approach to extract data records in Section 3. We give extensive experimental evaluations in Section 4 and conclude in Section 5.

2 Related Work

As mentioned, data record extraction has been extensively studied for structured and semi-structured resources (e.g., Muslea et al., 2001; Arasu and Garcia-Molina, 2003; Liu et al., 2003; Zhu et al., 2006). Most of those approaches rely on the analysis of document structure (reflected in, for example, html tags), from which record templates are derived. However, this approach does not apply to unstructured text. The reason lies in the difficulty of representing a data record template in free text without formatting tags and integrating it into a learning system. We show how to address this problem by deriving data record templates through language analysis and representing them with a discourse level CRF model.

Given the problem of identifying one or more records in free text, it is natural to turn toward text segmentation. The Natural Language Processing (NLP) community has come up with various solutions towards topic-based text segmentation (e.g., Hearst, 1994; Choi, 2000; Malioutov and Barzilay, 2006). Most unsupervised text segmentation approaches work under optimization criteria to maximize the intra-segment similarity and minimize the inter-segment similarity based on word distribution statistics. However, this approach cannot be applied directly to data record extraction. A careful study of our corpus shows that data records share many words and phrases and are not distinguishable based on word similairties. In other words, different experiments (records) always belong to the same topic and there is no way to segment them using standard topic segmentation techniques (even if one views the problem as a finer-level segmentation than traditional text segmentation). In addition, most text segmentation approaches require a prespecified number of segments, which in our domain cannot be provided.

(Wick et al., 2006) report extracting database records by learning record field compatibility. However, in our case, the field compatibility is hard to distinguish even by a human expert. Cluster-based or pairwise field similarity measures do not apply to our corpora without complex knowledge reasoning. Most of Wick et al.'s data (faculty and student's homepages) contains one record.

In addition, as explained below, we have found that surface word statistics alone are not sufficient to derive data record templates for extraction. Some (limited) form of semantic understanding of text is necessary. We therefore first perform some sentence level extraction (following the HP problem) and then integrate semantic labels and semantic language model features into a discourse level CRF model to represent the template for extracting data records in the future.

Recently an increasing number of research efforts on text mining and IE have used CRF models (e.g., Peng and McCallum, 2004). The CRF model provides a compact way to integrate different types of features when sequential labeling is important. Recent work includes improved model variants (e.g., Jiao et al., 2006; Okanohara et al., 2006) and applications such as web data extraction (Pinto et al., 2003), scientific citation extraction (Peng and McCallum, 2004), and word alignment (Blunsom and Cohn, 2006). But none of them have used CRFs for discourse level data record extraction.

We use a CRF model to represent a data record template and integrate various knowledge as CRF features. Instead of traditional work on the sentence level, our focus here is on the discourse level. As this has not been carefully explored, we experiment with various selected features.

For the biomedical domain, our work will facilitate biomedical research by supporting the construction of Knowledge Base Management Systems (e.g., Stephan et al., 2001; Hahn et al., 2002; Burns and Cheng, 2006). Unlike the well-studied problem of relation extraction from biomedical text, our work focuses on grouping extracted attributes across sentences into meaningful data records. TTE experiment is only one of many experimental types in biology. Our work can be generalized to many different types of data records to facilitate biology research.

In the next section, we present our approach to extracting data records.

3 Extracting Data Records

Inspired by the idea of Noun Phrase (NP) chunking in a single sentence, we view the data records extraction problem as discourse chunking from a sequence of sentences using a sequential labeling CRF model.

3.1 Sequential Labeling Model: CRF

The CRF model addresses the problem of labeling sequential tokens while relaxing the strong independence assumptions of Hidden Markov Models (HMMs) and avoiding the presence of label bias from having few successor states. For each current state, we obtain the conditional probability of its output states given previously assigned values of input states. For most language processing tasks, this model is simply a linear-chain Markov Random Fields model.

In typical labeling processes using CRFs each token is viewed as a labeling unit. For our problem, we process each input document $D=(s_1,s_2,...,s_n)$ as a sequence of individual sentences, with a corresponding labeling sequence of labels, $L = (l_1, l_2, ..., l_n)$, so that each sentence corresponds to only one label. In our problem, each data record corresponds to a distinct TTE experiment. Similar to NP chunking, we define three labels for sentences, "B_REC" (beginning of record), "I_REC" (inside record), and "O" (other). The default label "O" indicates that this sentence is beyond our concern.

The CRF model is trained to maximize the probability of $P(L \mid D)$, that is, given an input document *D*, we find the most probable labeling sequence *L*. The decision rule for this procedure is:

$$\hat{L} = \arg\max_{L} P(L \mid D) \qquad (1)$$

A CRF model of the two sequences is characterized by a set of feature functions f_k and their corresponding weights λ_k. As in Markov fields, the conditional probability $P(L \mid D)$ can be computed using Equation 2.

$$P(L \mid D) = \frac{1}{Z_S} \exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k * f_k(l_{t-1}, l_t, D, t) \right) \qquad (2)$$

where $f_k(l_{t-1}, l_t, D, t)$ is a feature function, representing either the state transition feature $f_k(l_{t-1}, l_t, D)$ or the feature of output state $f_k(l_t, D)$ given the input sequence. All these feature functions are user-defined boolean functions.

CRF works under the framework of supervised learning, which requires a pre-labeled training set to learn and optimize system parameters to maximize the probability or its log format. Equipped with this model, we investigate how to apply it and prepare features accordingly.

3.2 Feature Preparation

The CRF model provides a compact, unified framework to integrate features. However, unlike sentence-level processing, where features are very intuitive and circumscribed, it is not obvious what features are most indicative for our problem. We therefore explore three categories of features for discourse level chunking.

3.2.1 Semantic Attribute Labels

Most text segmentation approaches compute surface word similarity scores in given corpora without semantic analysis. However, in our case, data records have very similar characteristics and share most of the words. They are not distinguishable just from an analysis of surface word statistics. We have to understand the semantics before we can make decisions about data record extraction.

In our case, we care about the four types of attributes of each data record (one TTE experiment). Table 1 gives the definitions of the four attributes for each data record.

Name	Description
injectionLocation	the named brain region where the injection was made.
tracerChemical	the tracer chemical used.
labelingLocation	the region/location where the labeling was found.
labelingDescription	a description of labeling, including label density or label type.

Table 1. Attributes of data records (a TTE experiment).

To obtain this semantic attributes information of individual sentences (the HP problem), we first apply another sentence-level CRF model to label each sentence. We consider five categories of features based on language analysis. Table 2 shows the features for each category.

Name	Feature	Description
Lexicon Knowledge	TOPOGRAPHY	Is word topographic?
	BRAIN_REGION	Is word a region name?
	TRACER	Is word a tracer chemical?
	DENSITY	Is word a density term?
	LABELING_TYPE	Does word denote a labeling type?
Surface Word	Word	Current word
Context Window	CONT-INJ	If current word is within a window of injection context
Window Words	Prev-word	Previous word
	Next-word	Next word
Dependency Features	Root-form	Root form of the word if different
	Gov-verb	The governing verb
	Subject	The sentence subject
	Object	The sentence object

Table 2. The features for labeling words.

a. **Lexicon knowledge**. We used names of brain structures taken from brain atlases (Swanson, 2004), standard terms to denote neuroanatomical topographical relationships (e.g., “rostral”), the name or abbreviation of the tracer chemical used (e.g., “PHAL”), and commonsense descriptions for descriptions of the labeling (e.g., “dense”, “light”).

b. **Surface and window word**. The current word and the words around are important indicators of the most probable label.

c. **Context window.** The TTE is a description of the inject-label-findings process. Whenever a word having a root form of “injection” or “deposit” appears, we generate a context window and all the words falling into this window are assigned a feature of “CONT-INJ”.

d. **Dependency features**. We apply a dependency parser MiniPar (Lin, 1998) to parse each sentence, and then derive four types of features from the parsing result. These features are (a) root form of every word, (b) the subject within the sentence, (c) the object within the sentence, and (d) the governing verbs.

The labeling system assigns a label for every token in each sentence. We achieved the best performance with an F-score of 0.79 (based on a precision of 0.80 and a recall of 0.78). This is not the focus of this paper. Please refer to our previous work (Burns et al., 2007) for details.

```
<SENT FILE="1995-360-213-ns.xml" INDEX= "63">
Regardless of the precise location of <tracerChemical>
PHAL </tracerChemical> injection sites in <injectionLo-
cation> the MEA </injectionLocation> , <labelingDe-
scription> labeled axons </labelingDescription> followed
the same basic routes .
</SENT>
```

Figure 2. An example of semantic attribute labels.

With the sentence-level understanding of each sentence, we obtain the semantic attribute labels for the data records. Figure 2 gives an example sentence with semantic attribute labels. Here <tracerChemical>, <labelingLocation>, and <labelingDescription> are recognized by the system, and the attribute names will be used as features for this sentence.

3.2.2 Semantic Language Model

Since text narratives might adhere to logical ways of expressing facts, language models for each sentence will also provide good features to extract data records. However, in biomedical research articles many of the technical words/phrases used in the narrative are repeated across experiments, making the surface word language model of little use in deriving generalized data record templates. Considering this, we replace in each sentence the labeled fragments with their attribute labels and then derive semantic language models from that format. By ‘semantic language model’ we therefore mean a combination of semantic labels and surface words.

For example, in the sentence shown in Figure 2, we have the semantic language model trigrams location-of-<tracerChemical>, sites-in-<injectionLocation>, and <labelingDescription>-followed-the. In addition, we also query WordNet for the root form of each word to generalize the semantic language models. This for example produces the semantic language model trigrams site-in-<injectionLocation> and <labelingDescription>-follow-the.

We believe the collected semantic language models represent an inherent structure of unstructured data records. By integrating them as features with a CRF model, we expect to represent data record templates and use the learned model to extract new data records.

However, it is not clear what semantic language models are most indicative and useful. A bag-of-words (language models) approach may bring much noise in. We show below a comparison of regular language models and semantic language models in evaluations.

3.2.3 Layout and Word Heuristics

The previous two categories of features come from the discovery of semantic components of sentences and their narrative form word analysis. When interviewing the neuroscience expert annotator, we learned that some layout and word level heuristics may also help to delineate individual data records. Table 3 gives the two types of heuristic features.

When a sentence contains heuristic words, it will be assigned to a word heuristic feature. If the sentence is at the boundary of a paragraph, it will be assigned a layout heuristic feature, namely the first or the last sentence in the paragraph.

Name	Feature	Description
EXP_B_WORD	INJECT CASE EXPERIMENT APPLICATION DEPOSIT PLACEMENT INTRODUCTION	Heuristic words for beginning of an experiment description
POS_IN_PARA	FIRST_IN_PARA LAST_IN_PARA	Position of the sentence in the paragraph

Table 3. The heuristic features.

4 Empirical Evaluation

To evaluate the effectiveness and performance of our technique, we conducted extensive experiments to measure the data record extraction approach.

4.1 Experimental Setup

We used the machine learning package MALLET (McCallum, 2002) to conduct the CRF model training and labeling.

We have obtained the digital publications of 9474 *Journal of Comparative Neurology (JCN)*[1] articles from 1982 to 2005. We have converted the PDF format into plain text, maintaining paragraph breaks (some errors still occur though). A simple heuristic based approach identifies semantic sections of the paper (e.g, Introduction, Results, Discussion). As most experimental descriptions appear in the Results section, we only process the Results section. A neuroscience expert manually annotated the data records in the Results section of 58 research articles. The total number of sentences in the Results section of the 58 files is 6630 (averaging 114.3 sentences per article).

	Training Set	Testing Set
Docs	39	19
Data Records	249	133

Table 4. Experiment configuration.

We randomly divided this material into training and testing sets under a 2:1 ratio, giving 39 documents in the training set and 19 in the testing set. Table 4 gives the numbers of documents and data records in the training and the testing set.

4.2 Evaluation Metrics

To evaluate data record extraction, we notice it is not fair to strictly evaluate the boundaries of data records because this does not penalize the near-miss and false positive of data records in a reasonable way; sentences near a boundary that contain no relevant record information can be included or omitted without affecting the results. Hence the standard P_k (Beeferman et al., 1997) and *WinDiff* (Pevzner and Hearst, 2002) measures for text segmentation are not so suitable for our task.

As we are concerned with the usefulness of knowledge in extracted data records, we instead evaluate from the perspective of IE. We measure system performance on the quality of the extracted data records. For each extracted data record, it will be aligned to one of the data records in the gold standard using the "dominance rule" (if the data record can be aligned to multiple records in the gold standard, it will be aligned to the one with highest overlap). Then we evaluate the precision, recall, and F1 scores of extracted units of the data record. The units are the attributes in data records.

$$precision = \frac{\# of \text{ correct units}}{\# \text{of the extracted units by the system}} \quad (3)$$

$$recall = \frac{\# of \text{ correct units}}{\# \text{of the units in the gold standard}} \quad (4)$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \quad (5)$$

These measures provide an indication of the completeness and correctness of each extracted record (experiment). We also measure the number of distinct records extracted, compared with the gold standard as appearing in the document.

4.3 Experiment Results

To fully compare the effectiveness of our semantic analysis functionality, we evaluated system performance for all the following systems:

TextTiling (TT): To compare with text segmentation techniques, we use TextTiling (Hearst, 1994) with default parameters as the first baseline system.

Random Guess (RG): In order to demonstrate the data balance of all the possible labels in the testing set, we also use another baseline system with random decisions for each sentence.

[1] http://www3.interscience.wiley.com/cgi-bin/jhome/31248

Domain Heuristics (DH): In a regular TTE experiment, only one tracer chemical will typically be used. Given this heuristic, we assume each data record contains one tracer chemical. In this system, we first locate sentences with identified trace chemicals, and then we greedily expand backward and forward until another new tracer chemical appears or no other attribute is included.

Surface Text (ST): To measure the effectiveness of the semantic analysis (attribute labels and semantic language models), the ST system utilizes only standard surface word language models and heuristic features.

Semantic Analysis (SEM): The SEM system uses all the semantic features available (including identified attributes and semantic language models) and two heuristic features.

Table 5 shows the final performance of these different systems. The second column provides the numbers of extracted data records. In this task, a larger number does not necessarily mean a better system, as a system might produce too many false positives. The remaining three columns represent the precision, recall, and F1 scores, averaged over all data records. With our approach, the system performance is significantly improved compared with other systems. System TT fails in this task as it only outputs the full document as one single record.

	# of Records	Prec.	Rec.	F1
TT	19	0.3861	1.0	0.5571
RG	758	0.6331	0.0913	0.1595
DH	162	0.6703	0.4902	0.5663
ST	82	0.8182	0.8339	0.8260
SEM	72	**0.8505**	**0.9258**	**0.8865**

Table 5. System performance.

To investigate how plain text language models and semantic language models affect system performance, we also experimented with all the language models. Table 6 shows comparisons of three types of language models. Systems with semantic analysis always work better than those with only surface text analysis. Without semantic analysis, unigram features work better than bigram and trigram features. This matches our intuition: without generalizing to semantic language models, higher order language models will be relatively sparse and contain much noise. However, when taking into account the semantic features, we found that bigram and trigram semantic language model features outperformed unigrams. They are especially important in boosting the recall scores as they capture more generalized information when derived.

	Unigram (%)	Bigram (%)	Trigram (%)
	Prec/Rec/F1	Prec/Rec/F1	Prec/Rec/F1
ST	**81.8/83.4/82.6**	69.1/88.4/77.6	57.9/88.8/70.1
SEM	85.1/86.6/85.6	**85.1/92.6/88.7**	82.2/92.7/87.1

Table 6. Language model comparisons.

As an example, Table 7 gives a list of high quality bigram semantic language models ranked by their information gains based on the training data.

through_<labelingLocation>	rat_no
<labelingDescription>_be	of_<tracerChemical>
<labelingLocation>_(	<tracerChemical>_be
<tracerChemical>_injection	be_inject
into_<injectionLocation>	be_center
<labelingDescription>_from	inject_with
<tracerChemical>_in	injection_of
in_<labelingLocation>	in_experiment

Table 7. An example list of top-ranked bigrams.

The main difficulty for data record extraction from unstructured text lies in deriving and representing a template for future extraction. We actually take advantage of CRF and represent the template with a CRF model.

Each data record is measured with precision, recall, and F1 scores. Figure 3 depicts the distribution of extracted data records according to these measures in the best system.

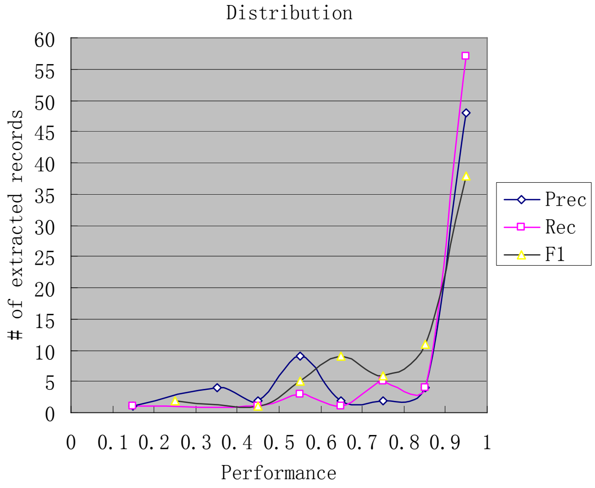


Figure 3. Data records performance distribution.

The results are encouraging, especially given the complexity and flexibility of data record descriptions in the unstructured text. In Figure 3, Axis X represents the value interval for precision, recall, and F1, and Axis Y represents the number of extracted records with their corresponding values. For example, 57 records have recall scores falling into [0.9, 1.0].

Figure 4 gives an example alignment between system result and the gold standard. Each record is represented by a range of sentences. The numbers following each record in the system result are individual data record's precision and recall scores.

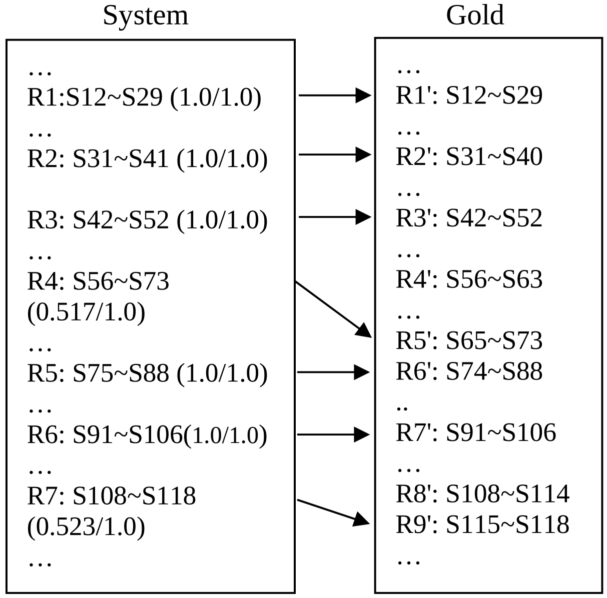


Figure 4. An example of record extraction in one doc.

This is a real example from the testing set. For records R1, R3, and R6, the system can extract the exact sentences contained. For record R2 and R5, although they do not exactly match at the sentence level, the extracted record contains the entire required set of attributes as in the gold standard.

4.4 Error Analysis and Discussion

When we investigated the errors, we found that sometimes the extracted data records combined two or more smaller gold standard records, or vice versa. As shown in Figure 4, extracted records R4 and R7 are both combinations of records in the gold standard. This is partially due to the granularity definition problem. Authors may mention several approaches/symptoms to one type of experiment for a single purpose. In this case, it is almost infeasible to have annotators strictly agree on granularity and thus to teach the system to acquire this knowledge. For example, in the gold standard, the annotator annotated three successive sentences as three separate records but the system output those as only one data record. In this extreme case, it is too hard to expect the system to perform well.

In our approach, the semantic attribute labels and semantic language models require the result of the initial sentence-level labeling, which has an F-score of 0.79. The error may propagate into the data record extraction procedure and lower overall system performance.

In our current experiments, we also assume all the attributes within one segment belong to one record. However, the situation of embedded data records will make this problem harder. For example, authors sometimes compare the current experiment with other approaches in referenced papers. In this case, those attributes should be excluded from the records. We need to invent rules or constraints to filter them out. When such reference occurs at experiment boundaries, it brings higher risk for correct results.

It is a very hard problem to extract from unstructured text neat structured records. The annotators sometimes employ background knowledge or reasoning when performing manual extraction; such knowledge cannot today be easily modeled and integrated into learning systems.

In our study, we also compared some feature selection approaches. Similar to (Yang and Pedersen, 1997), we tried Feature Instance Frequency, Mutual Information, Information Gain, and CHI-square test. But we eventually found that the system including all the features worked best, and with all the other configurations unchanged, feature instance frequency worked at almost the same level as other complex measures such as mutual information and information gain.

5 Conclusion and Future Work

In this paper, we explored the problem of extracting data records from unstructured text. The lack of structure makes it difficult to derive meaningful objects and their values without resorting to deeper language analysis techniques. We derived indicative linguistic features to represent data record templates in free text, using a two-pass approach in which the second pass used the IE labels derived from the first to compose attributes into coherent data records. We evaluated the results from an IE perspective and reported potential problems of error generation.

For the future, we plan to explore additional feature types and feature selection strategies to determine what is "good" for unstructured record templates to improve our results. More effort will also be put into the sentence-level analysis to reduce error propagations. In addition, ontology based knowledge inference strategies might be useful to validate attributes in single record and in turn help data record extraction. The last thing under our direction is to explore new models if applicable.

We hope this thought-provoking problem will attract more attention from the community. In the future, we plan to make our corpus available to the community. The solution to this problem will highly affect the access of knowledge in large scale unstructured text corpora.

Acknowledgements

The work was supported in part by an ISI seed funding, and in part by a grant from the National Library of Medicine (RO1 LM07061). The authors want to thank Feng Pan for his helpful suggestions with the manuscript. We would also like to thank the anonymous reviewers for their valuable comments.

References

Arasu, A., and Garcia-Molina, H. 2003. Extracting structured data from web pages. In *Proc. of SIMOD-2003*.

Beeferman, D., Berger, A., and Lafferty, J. 1997. Text segmentation using exponential models. In *Proc. of EMNLP-1997*.

Blunsom, P. and Cohn, T. 2006. Discriminative word alignment with conditional random fields. In *Proc. of ACL-2006*.

Brazma, A., et al., 2001. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet*, 29(4): p. 365-71.

Burns, G.A. and Cheng, W.-C. 2006. Tools for knowledge acquisition within the NeuroScholar system and their application to anatomical tract-tracing data. In *Journal of Biomedical Discovery and Collaboration*.

Burns, G., Feng, D., and Hovy, E.H. 2007. Intelligent Approaches to Mining the Primary Research Literature: Techniques, Systems, and Examples. Book Chapter in *Computational Intelligence in Bioinformatics,* Springer-Verlag, Germany**.**

Choi, F. Y. Y. 2000. Advances in domain independent linear text segmentation. In *Proc. of NAACL-2000*.

Hahn, U., Romacher, M., and Schulz, S. 2002. Creating knowledge repositories from biomedical reports the MEDSYNDIKATE text mining system. In *Proc. of PSB-2002*.

Hearst, M. 1994. Multi-paragraph segmentation of expository text. In *Proc. of ACL-1994*.

Jiao, F., Wang, S., Lee, C., Greiner, R., and Schuurmans, D. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proc. of ACL-2006*.

Kristjannson, T., Culotta, A. Viola, P., and McCallum, 2004. A. Interactive information extraction with constrained conditional random fields. In *Proc. of AAAI-2004*.

Lafferty, J., McCallum, A. and Pereira, F. 2001 Conditional Random Fields: probabilistic models for segmenting and labeling Sequence Data. In *Proc. of ICML-2001*.

Lin, D. 1998. Dependency-based evaluation of MINIPAR. In *Proc. of Workshop on the Evaluation of Parsing Systems*.

Liu, B., Grossman, R., and Zhai, Y. 2003. Mining data records in web pages. In *Proc. of SIGKDD-2003*.

Malioutov, I. and Barzilay, R. 2006. Minimum cut model for spoken lecture segmentation. In *Proc. of ACL-2006.*

McCallum, A.K. 2002. *MALLET: A Machine Learning for Language Toolkit*. http://mallet.cs.umass.edu.

Muslea, I., Minton, S., and Knoblock, C.A. 2001. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems* 4:93-114.

Okanohara, D., Miyao, Y., Tsuruoka, Y., and Tsujii, J. 2006. Improving the scalability of semi-markov conditional random fields for named entity recognition**.** In *Proc. of ACL-2006*.

Peng, F. and McCallum, A. 2004. Accurate information extraction from research papers using conditional random fields. In *Proc. of HLT-NAACL-2004*.

Pevzner, L., and Hearst, M. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*.

Pinto, D., A. McCallum, X. Wei, and W.B. Croft. 2003. Table Extraction Using Conditional Random Fields. In *Proc. of SIGIR-2003*.

Stephan, K.E. et al., 2001. Advanced database methodology for the Collation of Connectivity data on the Macaque brain (CoCoMac). *Philos Trans R Soc Lond B Biol Sci*, 356(1412).

Swanson, L.W. 2004. *Brain Maps: Structure of the Rat Brain*. 3rd edition, Elsevier Academic Press.

Wick, M., Culotta, A., and McCallum, A. 2006. Learning field compatibilities to extract database records from unstructured text. In *Proc. of EMNLP-2006.*

Yang, Y., and Pedersen, J. 1997. A comparative study on feature selection in text categorization. In *Proc. of ICML-1997*, pp. 412-420.

Zhu, J., Nie, Z., Wen, J., Zhang, B., and Ma, W. 2006. Simultaneous record detection and attribute labeling in web data extraction. In *Proc. of KDD-2006*.

Multiple Alignment of Citation Sentences with Conditional Random Fields and Posterior Decoding

Ariel S. Schwartz[*]
EECS, Computer Science Division
UC Berkeley
Berkeley, CA 94720-1776
sariel@cs.berkeley.edu

Anna Divoli, Marti A. Hearst
School of Information
UC Berkeley
Berkeley, CA 94720-4600
{hearst,divoli}@ischool.berkeley.edu

Abstract

In scientific literature, sentences that cite related work can be a valuable resource for applications such as summarization, synonym identification, and entity extraction. In order to determine which equivalent entities are discussed in the various citation sentences, we propose *aligning* the words within these sentences according to semantic similarity. This problem is partly analogous to the problem of multiple sequence alignment in the biosciences, and is also closely related to the word alignment problem in statistical machine translation. In this paper we address the problem of multiple citation concept alignment by combining and modifying the CRF based pairwise word alignment system of Blunsom & Cohn (2006) and a posterior decoding based multiple sequence alignment algorithm of Schwartz & Pachter (2007). We evaluate the algorithm on hand-labeled data, achieving results that improve on a baseline.

1 Introduction

The scientific literature of biomedicine, genomics, and other biosciences is a rich, complex, and continually growing resource. With appropriate information extraction and retrieval tools, bioscience researchers can use the contents of the literature to further their research goals. With online full text of journal articles finally becoming the norm, new forms of citation analysis become possible.

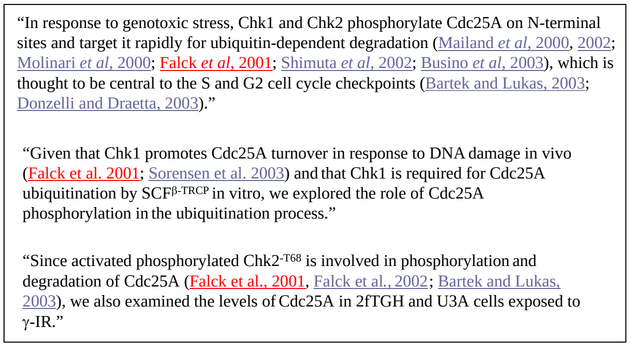
Nearly every statement in biology articles is backed up by at least one citation, and, conversely, it is quite common for papers in the bioscience domain to be cited by 30–100 other papers. The cited facts are typically stated in a more concise way in the citing papers than in the original papers. Since the same facts are repeatedly stated in different ways in different papers, statistical models can be trained on existing citation sentences to identify similar facts in unseen text. Citation sentences also have the potential to be useful for text summarization and database curation. Figure 1 shows an example of three different citation sentences to the same target paper.

Most citation analysis work focuses on the citation network structure, to determine which papers are most central, or uses co-citation analysis to determine which papers are similar to one another in content (White, 2004; Liu, 1993; Garfield, 1955; Lipetz, 1965; Giles et al., 1998). In this paper we focus instead on analyzing the sentences that surround the citations to related work, which we termed *citances* in Nakov et al. (2004). In that paper we note that one subproblem for using citances for automated analysis is to identify the different concepts mentioned; a given paper may be cited for more than one fact or relation.

Citances often state similar information using varying words and phrases. In order to build concise summaries, those entities and relations that are expressed in different ways should be matched up, or *aligned*, so that subsequent processing steps will know what the core concepts are. In this paper we

[*] Current address: Department of Bioengineering, University of California, San Diego, La Jolla, CA 92093-0412. Email: sariel@ucsd.edu.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 847–857, Prague, June 2007. ©2007 Association for Computational Linguistics

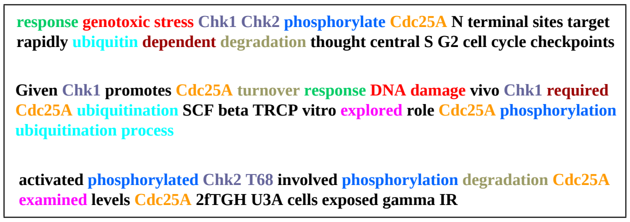


"In response to genotoxic stress, Chk1 and Chk2 phosphorylate Cdc25A on N-terminal sites and target it rapidly for ubiquitin-dependent degradation (Mailand *et al*, 2000, 2002; Molinari *et al*, 2000; Falck *et al*, 2001; Shimuta *et al*, 2002; Busino *et al*, 2003), which is thought to be central to the S and G2 cell cycle checkpoints (Bartek and Lukas, 2003; Donzelli and Draetta, 2003)."

"Given that Chk1 promotes Cdc25A turnover in response to DNA damage in vivo (Falck et al. 2001; Sorensen et al. 2003) and that Chk1 is required for Cdc25A ubiquitination by SCF$^{\beta\text{-TRCP}}$ in vitro, we explored the role of Cdc25A phosphorylation in the ubiquitination process."

"Since activated phosphorylated Chk2$^{\text{-T68}}$ is involved in phosphorylation and degradation of Cdc25A (Falck et al., 2001, Falck et al., 2002; Bartek and Lukas, 2003), we also examined the levels of Cdc25A in 2fTGH and U3A cells exposed to γ-IR."

Figure 1: **Example of three unaligned citances.**



response genotoxic stress Chk1 Chk2 phosphorylate Cdc25A N terminal sites target rapidly ubiquitin dependent degradation thought central S G2 cell cycle checkpoints

Given Chk1 promotes Cdc25A turnover response DNA damage vivo Chk1 required Cdc25A ubiquitination SCF beta TRCP vitro explored role Cdc25A phosphorylation ubiquitination process

activated phosphorylated Chk2 T68 involved phosphorylation degradation Cdc25A examined levels Cdc25A 2fTGH U3A cells exposed gamma IR

Figure 2: **Example of three normalized aligned citances.** Homologous entities are colored the same. Unaligned entities are black.

build on the work of Nakov et al. (2004) by tackling the entity normalization step.

The citance alignment problem is partially analogous to the problem of multiple alignment of biological sequences (Durbin et al., 1998). In both cases the goal is to *align* homologous entities that are derived from the same ancestral entity. While in biology homology is well-defined in the molecular level, in the citances case it is defined in the semantic level, which is much more subjective. Given a group of citances that cite the same target paper, we loosely define *semantic homology* as a symmetric, transitive, and reflexive relation between two entities (words or phrases) in the same or different citance that have similar semantics in the context of the cited paper.

Figure 1 shows an example of three citances that cite the same target paper (Falck et al., 2001). A multiple alignment of the entities in the same citances (after removal of stopwords) is shown in Figure 2. Homologous entities are colored the same. This small example illustrates some of the main challenges of *multiple citance alignment* (MCA). While orthographic similarity can help to identify semantic homology (e.g., *phosphorylate* and *phosphorylation*), it can also be misleading (e.g., *cell cycle* and *U3A cells*). In addition, semantic homology might not include any orthographic clues (e.g., *genotoxic stress* and *DNA damage*).

Unlike global multiple sequence alignment (MSA) in genomics, where each character can be aligned to at most one character in every other sequence, in multiple citance alignment, each word can be aligned to any number of words in other sentences. Another major difference between the two problems is the fact that while the sequential ordering of characters must be maintained in multiple sequence alignment, this is not the case for multiple citance alignment.

MCA is also related to the problem of word alignment in statistical machine translation (SMT) (Och and Ney, 2003). However, unlike SMT alignment, MCA aligns multiple citances in the same language rather than a pair of sentences in different languages.

In this paper we present an MCA algorithm that is based on an extension to the posterior decoding algorithm for MSA called AMAP (Schwartz et al., 2006; Schwartz and Pachter, 2007), with an underlying pairwise alignment model based on the CRF SMT alignment of Blunsom & Cohn (2006).

2 Multiple citance alignments

Let $\mathcal{G} \triangleq \{C^1, C^2, \ldots, C^K\}$ be a group of K citances that cite the same target paper, where the i^{th} citance is a sequence of words $C^i \triangleq C^i_1 C^i_2 \cdots C^i_{n^i}$, and $c^i \triangleq \{c^i_1, c^i_2, \ldots, c^i_{n^i}\}$ is the set of word indices of C^i. A *pairwise citance alignment* of C^i and C^j is an equivalence (symmetric, reflexive, and transitive) relation $\sim_{ij}$ on the set $c^i \cup c^j$. The expression $c^i_k \sim_{ij} c^j_l$ means that according to the pairwise alignment $\sim_{ij}$ word k in citance C^i and word l in citance C^j are aligned. A *multiple citance alignment* (MCA) is an equivalence relation $\sim \triangleq \left(\bigcup_{ij} \sim_{ij}\right)^+$ on the set $\bigcup_i c^i$, which is the transitive closure of the union of all pairwise alignments of citance pairs in $\mathcal{G}$. Taking the transitive closure and not only the union of all pairwise alignments ensures that the MCA is an equivalence relation.

An MCA $\sim$ defines a partition of the set of all word indices $c \triangleq \bigcup_{ik} \{c^i_k\}$, which is of size $n \triangleq$

$|c| = \sum_i n^i$. Therefore, the number of distinct MCAs of $\mathcal{G}$ is the number of partitions of a set of size n. This number is called the n^{th} Bell number (Rota, 1964)

$$B_n \triangleq \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}. \tag{1}$$

Asymptotically, B_n grows faster than an exponential but slower than a factorial. For example $B_{100} \approx 10^{116}$. Obviously, enumerating all possible MCAs is impractical even for small problems.

3 Probabilistic model for MCA

Unlike biological sequences, for which pair-HMMs are a natural choice for modeling evolutionary processes between two sequences, there is no simple generative model that can be used for modeling pairwise citance alignment. Most of the work on pairwise alignment of sentences at the word level has been done in the statistical machine translation (SMT) community.

Och & Ney (2003) present an overview and comparison of the most common models used for SMT word alignments. Out of the models they describe, the HMM models are the most expressive models that can compute posterior probabilities using the forward-backward algorithm. However, unlike sequence alignments, there are no ordering constraints in word alignments, and the alignments are many-to-many as opposed to one-to-one. Therefore, the SMT HMM models cannot be based on pair-HMMs, which generate two sentences simultaneously. Rather, they are directional models that model the probability of generating a target sentence given a source sentence. In other words they only model many-to-one alignments, recovering the many-to-many alignments in a preprocessing step. Therefore, SMT HMMs can only compute the posterior probabilities $P(c^i_k \rightsquigarrow c^j_l|C^i, C^j)$ and $P(c^j_l \rightsquigarrow c^i_k|C^i, C^j)$, where the relation $\rightsquigarrow$ represents the (directional) event that a source word is translated into a target word. Nevertheless, recently such posterior probabilities have been used in SMT word alignment system as an alternative to Viterbi decoding, and helped to improve the performance of such systems (Matusov et al., 2004; Liang et al., 2006).

Generative models like HMMs have several limitations. First, they require relatively large training data, which is difficult to attain in case of SMT word alignment, and even more so in the case of MCA. Second, generative models explicitly model the inter-dependence of different features, which reduces the ability to incorporate multiple arbitrary features into the model. Since orthographic similarity is not a strong enough indication for semantic homology in MCA, we would like to be able to incorporate multiple inter-dependent features into a single model, including orthographic, contextual, ontological, and lexical features.

Recently, several authors have described discriminative SMT alignment models (Moore, 2005; Lacoste-Julien et al., 2006; Blunsom and Cohn, 2006). However, to the best of our knowledge only the model of Blunsom & Cohn (2006), which is based on a Conditional Random Field (CRF) (Lafferty et al., 2001), can compute word indices pairs' directional posterior probabilities, like those computed by the HMM models. Therefore, we decided to adopt the CRF-based model to the MCA problem.

3.1 Conditional random fields for word alignment

The model of Blunsom & Cohn (2006) is based on a linear chain CRF, which can be viewed as the undirected version of an HMM. The CRF models a many-to-one pairwise alignment, in which every source word can get aligned to zero or one target words, but every word in the target sentence can be the target of multiple source words. CRFs define a conditional distribution over a latent labeling sequence given observation sequence(s). In the case of CRF for word alignment, the observed sequences are the source and target sentences (citances), and the latent labeling sequence is the mapping of source words to target word-indices. Given a source citance C^i of length n^i, and a target citance C^j of length n^j, the many-to-one alignment of C^i to C^j is the relation $\rightsquigarrow$. Since this is a many-to-one alignment, $\rightsquigarrow$ can be represented by a vector a of length n^i. The CRF models the probability of the alignment a conditioned on C^i and C^j as follows:

$$P_\Lambda(a|C^i, C^j) = \frac{\exp\left(\sum_t \sum_k \lambda_k f_k(t, a_{t-1}, a_t, C^i, C^j)\right)}{Z_\Lambda(C^i, C^j)}, \tag{2}$$

where $f \triangleq \{f_k\}$ are the model's features, $\Lambda \triangleq \{\lambda_k\}$ are the features' weights, and $Z_\Lambda(C^i, C^j)$ is the partition (normalization) function which is defined as:

$$Z_\Lambda(C^i, C^j) \triangleq \sum_a \exp\left(\sum_t \sum_k \lambda_k f_k(t, a_{t-1}, a_t, C^i, C^j)\right). \quad (3)$$

Parameters are estimated from fully observed data (manually aligned citances) using a maximum a posteriori estimate. The parameter estimation procedure is described in more details in the original paper. Blunsom & Cohn (2006) use Viterbi decoding to find an alignment of two sentences given a trained CRF model, $a^* \triangleq \mathrm{argmax}_a P_\Lambda(a|C^i, C^j)$. However, the posterior probabilities of the labels at each position can be calculated as well using the forward-backward algorithm:

$$P_\Lambda(c_l^i \leadsto c_k^j | C^i, C^j) = P_\Lambda(a_l = c_k^j | C^i, C^j) = \frac{\alpha_l(c_k^j|C^i, C^j)\beta_l(c_k^j|C^i, C^j)}{Z_\Lambda(C^i, C^j)} \quad (4)$$

where α_l and β_l are the forward and backward vectors that are computed with the forward-backward algorithm (Lafferty et al., 2001).

3.2 The posterior decoding algorithm for MCA

Ultimately, the success of an MCA algorithm should be judged by its effect on the success of the citance analysis systems that use MCAs as their input. However, measuring this effect directly is difficult, since high-level tasks such as summarization are difficult to evaluate objectively. More to the point, it is difficult to quantify the contribution of the MCA accuracy to the accuracy of the high-level system that uses it. A more practical alternative is to measure the accuracy of MCAs directly using a meaningful accuracy measure, under the simplifying assumption that there is a strong correlation between the measured MCA accuracy and the performance of the high-level application.

We argue that a useful utility function should be correlated (or even identical) to the accuracy measure used to evaluate the performance of an algorithm. In addition, the utility function should be easily decomposable, to enable direct optimization using posterior-decoding. Although any accuracy measure that is acceptable as a single performance measure can be used to guide the design of the utility function, metric-based accuracy measures have several noticeable advantages. First, a metric formalizes the intuitive notion of distance. Hence, an accuracy measure which is based on a metric follows the intuition that reducing the distance to the correct answer should increase the accuracy of the predicted answer. Therefore, defining a metric space for the objects of a given problem leads to a natural definition of accuracy. Another advantage of using a metric-based accuracy measure is the ability to provide bounds in the search space using the triangle inequality. For example, while searching for the answer with the optimal (metric-based) expected utility, a step of length x can only change the expected utility as well as the actual utility by at most $\pm x$ units. Examples of more complex bounds using metric loss functions are described in (Schlüter et al., 2005) and (Domingos, 2000).

Schwartz et al. (2006) define the *alignment metric accuracy* (AMA), which is a utility function for one-to-one MSA. Intuitively, AMA measures the fraction of characters that are aligned correctly according to the reference alignment, either to another character or to a gap (null). We extend the definition of AMA to the case of many-to-many MCA.

A good utility function for MCA should give partial credit to word positions that align to some of the correct word positions while penalizing for aligning to wrong word positions. To help define such a utility function we define the following. Let $m_\sim^{ij}(c_l^j) \triangleq \{c_k^i \in c^i | c_k^i \sim c_l^j\}$ be the set of all word positions in citance C^i that align to word position l in citance C^j according to MCA $\sim$. We can then define the following utility function for the MCA $\sim^p$ of the citance group $\mathcal{G}$ given a reference MCA $\sim^r$:

$$U_{AMA}(\sim^r, \sim^p) \triangleq \frac{\sum_{ijl|i \neq j} U_{set\_agreement}\left(m_{\sim^r}^{ij}(c_l^j), m_{\sim^p}^{ij}(c_l^j)\right)}{n(K-1)}, \quad (5)$$

where n is the number of word indices in $\mathcal{G}$, $K \triangleq |\mathcal{G}|$ is the number of citances in the group, and $U_{set\_agreement}$ is any utility function for agreement between sets that assigns values in the range $[0, 1]$.

$U_{set\_agreement}$ can be viewed as a "score" assigned to each word position based on the agreement between the two alignments with regards to the other word positions that align to it. Using a 0–1 loss as the set agreement score is equivalent to the original AMA. Other utility functions, such as Dice, Jaccard and Hamming can be used as $U_{set\_agreement}$. However, only metric-based utility functions will result in a metric-based U_{AMA} utility function. It is easy to see that $1 - U_{AMA}$ satisfies all the requirements of a metric, i.e., it is non-negative, equals zero if and only if $\sim^r = \sim^p$, symmetric, and obeys the triangle inequality, since if the triangle inequality holds for $1 - U_{set\_agreement}$, it must hold for a sum of $1 - U_{set\_agreement}$ values. (We refer the reader to Schwartz (2007) for a longer discussion of the properties of the different utility functions.) We define the AMA for MCA by setting the $U_{set\_agreement}$ to be the Braun-Blanquet coefficient (Braun-Blanquet, 1932), which is defined as:

$$U_{Braun-Blanquet}\left(m^{ij}_{\sim^r}(c^j_l), m^{ij}_{\sim^p}(c^j_l)\right) \triangleq \begin{cases} 1 & \text{if } m^{ij}_{\sim^r}(c^j_l) = \emptyset \\ & \text{and } m^{ij}_{\sim^p}(c^j_l) = \emptyset \\ \frac{|m^{ij}_{\sim^r}(c^j_l) \cap m^{ij}_{\sim^p}(c^j_l)|}{\max\left\{|m^{ij}_{\sim^r}(c^j_l)|, |m^{ij}_{\sim^p}(c^j_l)|\right\}} & \text{otherwise} \end{cases}. \tag{6}$$

Caillez & Kuntz (1996) show that the Braun-Blanquet coefficient is based on a metric.

As with the MSA case, a family of utility functions can be defined to enable control of the recall/precision trade-off. Unlike MSA, in the case of MCA two free parameters are needed, in order to have better control of the trade-off using posterior-decoding. In addition to a *gap-factor* that controls the threshold at which unaligned words start to get aligned, a *match-factor* is added to enable control of the number of word-positions each word aligns to. The result is the following utility function:

$$U_{\mu,\gamma}(\sim^r, \sim^p) \triangleq \frac{1}{n(K-1)} \sum_{ijl|i \neq j} \Bigg(\mu^{|m^{ij}_{\sim^p}(c^j_l)|} \frac{|m^{ij}_{\sim^r}(c^j_l) \cap m^{ij}_{\sim^p}(c^j_l)|}{\max\left\{|m^{ij}_{\sim^r}(c^j_l)|, |m^{ij}_{\sim^p}(c^j_l)|, 1\right\}} + \gamma \mathbf{1}\{m^{ij}_{\sim^r}(c^j_l) = m^{ij}_{\sim^p}(c^j_l) = \emptyset\} \Bigg), \tag{7}$$

where $\gamma \in [0, \infty)$ is a gap-factor, and $\mu \in (0, \infty)$ is a match factor. The neutral value for both parameters is 1. Increasing γ results in increased utility to sparser MCAs, while reducing γ increases the utility of denser alignments. However, in the case of MCA, the gap-factor only affects the first aligned word position, but it cannot affect the number of word positions each word is aligned to. The match-factor adds this functionality by rewarding MCAs that align words to multiple word positions when $\mu > 1$, and penalizing such MCAs when $\mu < 1$.

Given a group of K citances $\mathcal{G}$ and a trained CRF model, the goal of the MCA algorithm is to find the MCA $\sim^* \triangleq \operatorname{argmax}_{\sim^p} E_{\sim^t} U_{\mu,\gamma}(\sim^t, \sim^p)$ that maximizes the expected utility. Since searching the space of possible MCAs exhaustively is infeasible, we resort to a simple heuristic for predicting an MCA. Instead of searching for a global optimum, the predicted MCA is defined as the equivalence (symmetric transitive) closure of the union of multiple local optima. For each target word position c^j_l and every source citance C^i the combination of source word positions $c^i_\circ$ that maximize the expected set-agreement score of c^j_l is added to the predicted MCA. Formally, let $\mathcal{P}(c^i)$ be the power-set of c^i, then we define the predicted MCA as $\sim^p \triangleq \left(\rightsquigarrow^p \cup (\rightsquigarrow^p)^{-1}\right)^+$, where $\rightsquigarrow^p$ is defined as:

$$\rightsquigarrow^p \triangleq \bigcup_{ijl|i \neq j} \{c^j_l\} \times \operatorname*{argmax}_{c^i_\circ \in \mathcal{P}(c^i)} E_{m^{ij}_{\sim^t}(c^j_l)} \left(\mu^{|c^i_\circ|} \frac{|m^{ij}_{\sim^t}(c^j_l) \cap c^i_\circ|}{\max\left\{|m^{ij}_{\sim^t}(c^j_l)|, |c^i_\circ|, 1\right\}} + \gamma \mathbf{1}\{m^{ij}_{\sim^t}(c^j_l) = c^i_\circ = \emptyset\} \right). \tag{8}$$

The value of $\rightsquigarrow^p$ can be computed from the CRF directional posterior probabilities as follows:

$$\rightsquigarrow^p = \bigcup_{ijl|i \neq j} \{c^j_l\} \times \operatorname*{argmax}_{c^i_\circ \in \mathcal{P}(c^i)} \sum_{c^i_* \in \mathcal{P}(c^i)} P\left(m^{ij}_{\sim^t}(c^j_l) = c^i_*\right) \left(\mu^{|c^i_\circ|} \frac{|c^i_* \cap c^i_\circ|}{\max\{|c^i_*|, |c^i_\circ|, 1\}} + \gamma \mathbf{1}\{c^i_* = c^i_\circ = \emptyset\} \right), \tag{9}$$

and using an independence assumption we get:

$$\leadsto^{p} \approx \bigcup_{ijl|i\neq j} \{c_l^j\} \times \operatorname*{argmax}_{c_\circ^i \in \mathcal{P}(c^i)} \sum_{c_*^i \in \mathcal{P}(c^i)} \left(\prod_{c_k^i} \Big(P_\Lambda(c_k^i \leadsto c_l^j | C^i, C^j) \mathbf{1}\{c_k^i \in c_*^i\} + (1 - P_\Lambda(c_k^i \leadsto c_l^j | C^i, C^j)) \mathbf{1}\{c_k^i \notin c_*^i\} \Big) \right) \left(\mu^{|c_\circ^i|} \frac{|c_*^i \cap c_\circ^i|}{\max\{|c_*^i|, |c_\circ^i|, 1\}} + \gamma \mathbf{1}\{c_*^i = c_\circ^i = \emptyset\} \right). \tag{10}$$

Note that although the directional posterior probabilities are used to generate the predicted MCA, the result is a many-to-many alignment, since the union is done over all pairs of sequences in both directions. The calculation in Equation (10) can be computationally intensive in practice, as it requires $|\mathcal{P}(c^i)|^2 = 2^{2n^i}$ operations for each word position c_l^j and citance C^i. This can be overcome by restricting the combinations of source word positions (c_*^i and $c_\circ^i$) to include only the the top MAX_SOURCES source words with a minimum posterior probability of MIN_PROB to align to c_l^j ($P_\Lambda(c_k^i \leadsto c_l^j|C^i, C^j) \geq$ MIN_PROB). In our implementation we set MAX_SOURCES to 8 and MIN_PROB to 0.01. Additionally, the probabilities of each combination c_*^i can be calculated only once, since it is independent of $c_\circ^i$. This reduces the total computational complexity of calculating $\leadsto^p$ to $O\left(2^{16}(K^2 - K)\max_{n^i}\{n^i\}\right)$.

4 Data sets

Since citance alignment is a new task, we had to create our own evaluation and training sets. We restricted the domain of the target papers to molecular interactions, a domain which is actively researched in the biosciences text mining community (Hirschman et al., 2002). The biologist in our group annotated citances to 6 target papers. The training set consisted of 40 citances to 4 different target papers (10 citances each; we wanted to have variety in the training set). The development set consisted of 51 citances to the fifth target paper, and the test set contained 45 citances to the sixth target paper.

For each target paper we downloaded the full text of those papers citing it that were available in HTML format. The link structure of the cited references in the HTML documents allowed us to automatically extract citances to a given target paper. We defined a citance to be the full sentence that contains a citation to the target paper. Each citance was then tokenized, and normalized by removing all stopwords from a predefined list.

One goal of the annotation was to cover as much of the content of the citances as possible. Another goal was consistency; our biologist manually followed a small number of rules to determine semantic similarity. Distinct semantic units (words or phrases) were identified and given an annotation ID. Within each group of citances, words or phrases that share semantic similarity were annotated with the same ID.

Using the manually annotated citance groups, pairwise word alignments were generated for every source-target pair of citances from every group. That resulted in a training, development, and test sets of 180, 1275, and 990 pairwise alignments respectively.

Alignments that were used for development and testing were generated as many-to-many alignments. However, many-to-many alignments are not suitable for the training the many-to-one CRF alignment model. When a given source word c_k^i aligns to multiple words in the target citance, the CRF model chooses only one target word as a true positive, while incorrectly treating the other true positive target words as true negatives. To alleviate this problem, in such cases we replaced all true-positive target words other than the first with '*', thus forcing them to act as real true negatives for the purpose of training. This adjustment does not solve the inherent limitation of the CRF's many-to-one modeling of a many-to-many alignment, but it prevents learning incorrect weights for good features.

5 Feature engineering

The CRF alignment model can combine multiple overlapping features. We evaluated the effectiveness of different features by training models on the training set and evaluating their performance on the development set. We considered variations of features that were part of the original system of Blunsom & Cohn (2006), and also designed new features that are specific to the problem of MCA.

Orthographic features

We used the following orthographic features from the original system of Blunsom & Cohn (2006) (below all features are either Boolean indicator functions (b) or real valued (r)):

- (b) exact string similarity of source-target words;
- (b) every possible source-target pair of length 3 prefixes;
- (b) exact string match of length 3 prefixes;
- (b) exact string match of length 3 suffixes;
- (r) absolute difference in word lengths;
- (b) both words are shorter than 4 characters.

In addition, the following orthographic features were added: indicator that both words include capital letters, and normalized edit-similarity of the two words ($1 - \frac{edit\_distance(c_k^i, c_l^j)}{max\{|c_k^i|,|c_l^j|\}}$).

Markov features

We used the following Markov features from the original system:

- (r) absolute jump width ($abs(a_t - a_{t-1} - 1)$, which measures the distance between the target words of adjacent source words;
- (r) positive jump width ($max\{a_t - a_{t-1} - 1, 0\}$);
- (r) negative jump width ($max\{a_{t-1} + 1 - a_t, 0\}$);
- (b) transition from null aligned source-word to non-null aligned source-word;
- (b) transition from non-null aligned source-word to null aligned source-word;
- (b) transition from null aligned source-word to null aligned source-word.

In addition we added the following Markov features in order to model the tendency of certain words to be part of longer phrases:

- (b) source-word aligns to the same target-word as the previous source-word;
- (b) source-word aligns to the same target-word as the next source-word;
- (b) transition from non-null aligned source-word to non-null aligned source-word.

Sentence position: We included the relative sentence position feature from the original system, which is defined as $abs(\frac{a_t}{|c^j|} - \frac{t}{c^i})$. Although it was not expected to be relevant for MCA, since the citances are not expected to align along the diagonal, this feature slightly improved the performance of the development set.

Null: An indicator function for leaving a source-word unaligned was retained from the original system. This is an essential feature since without it the CRF tends to over-align words, and produces meaningless posterior probabilities.

Ontological features: Orthographic and positional features alone do not cover all cases of semantic homology. We therefore included features that are based on domain specific ontologies.

Using an automated script we mapped specific words and phrases in every citance to MeSH[1] terms, Gene identifiers from Entrez Gene,[2] UniProt,[3] and OMIM.[4] We then added features indicating when the source and target words are annotated with the same MeSH term or the same gene identifier. We tried numerous features that compare MeSH terms based on their distance in the ontology, and other features that indicate whether a word is part of a longer term. However, none of these feature were selected for the final system.

In addition to biological ontologies we added a feature for semantic word similarity between the source and target words, based on the Lin (1998) WordNet similarity measure.

6 Results

We modified the CRF alignment system of Blunsom & Cohn (2006) to support MCA by incorporating the posterior decoding algorithm from Section 3.2 into the existing system. The CRF model was trained using the features that were selected using the development set, on a dataset that included the training and development MCAs. All the performance results in this section are reported on the test set, which includes 990 pairs of citances ($45 \times 44/2$), with a total of 34188 words (8547×44). On average, 20% of the source words are aligned to at least one other target word in a given reference pairwise alignment. Since the union of all the pairwise alignments results in only a single test MCA, it is hard to make strong arguments about the performance

[1] http://www.nlm.nih.gov/mesh/

[2] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene

[3] http://www.pir.uniprot.org/

[4] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM

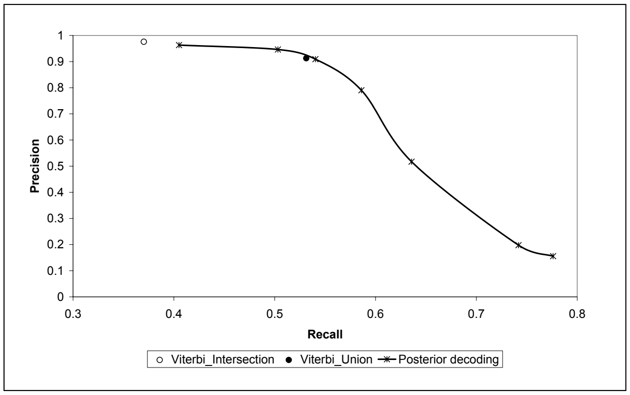


Figure 3: **Recall/Precision curve of pairwise citance alignments comparing Viterbi to posterior decoding.**

of the system in general. Therefore, we concentrate our discussion on general trends, and do not claim that the specific performance numbers we report here are statistically significant. As a point of comparison, the SMT community has been evaluating performance of word-alignment systems on an even smaller dataset of 447 pairs of non-overlapping sentences (Mihalcea and Pedersen, 2003).

We first analyze the performance of the system on pairwise citance alignments. Instead of taking the equivalence closure of $\rightsquigarrow^{p}$ we take only the symmetric closure. The result is 990 many-to-many pairwise alignments. In order to evaluate the effectiveness of the posterior-decoding algorithm, we generate the Viterbi alignments using the same CRF model. The Viterbi many-to-many pairwise alignments are then generated by combining equivalent pairs of many-to-one alignments using three different standard symmetrization methods for word-alignment—union, intersection, and the refined method of Och & Ney (2003).

Figure 3 shows the recall/precision trade-off of the pairwise posterior-decoding and Viterbi alignments. The curve for the posterior-decoding alignments was produced by varying the gap and match factors. For the Viterbi alignments, only three results could be generated (one for each symmetrization method). However, since the refined method produced a very similar result to the union, only the union is displayed in the figure. The important observation is that while posterior-decoding enables refined control over the recall/precision trade-off, the Viterbi decoding generates only three alignments, which cover only a small fraction of the curve at its high precision range. The union of Viterbi alignments achieves 0.531 recall at 0.913 precision, which is similar result to the 0.540 recall at 0.909 precision achieved using posterior-decoding with gap-factor and match-factor set to 1. However, unlike Viterbi, posterior-decoding produces alignments with much higher recall levels, by increasing the match-factor and decreasing the gap-factor. For example setting the gap-factor to 0.1 and match-factor to 1.2 results in alignments with 0.636 recall at 0.517 precision, and setting them to 0.05 and 1.5 results in 0.742 recall at 0.198 precision. Generally, the gap and match factor affect the accuracy of the alignments as expected. In particular, the alignments with the best AMA (0.889) and the best F_1-measure (0.678) are generated when the gap match factor are set to their natural values (1,1), which theoretically should maximize the expected AMA.

The performance of the pairwise alignments validates the underlying probabilistic model, showing it behaves as theoretically expected. However, the union of all pairwise alignments is not a valid MCA. To evaluate the MCA posterior decoding algorithm, we compared it to baseline MCAs. The baseline MCAs are constructed by using only the normalized-edit-distance $\frac{edit\_distance(c_k^i,c_l^j)}{\max\{|c_k^i|,|c_l^j|\}}$, and defining $c_k^i \rightsquigarrow^{\delta} c_l^j$ if and only if $normalized\_edit\_distance(c_k^i, c_l^j) \leq \delta$, where δ is a distance threshold. The final baseline MCA is constructed by taking the equivalence closure of all pairwise alignments, $\sim^{\delta} \triangleq \Big(\rightsquigarrow^{\delta} \cup (\rightsquigarrow^{\delta})^{-1})\Big)^{+}$. The δ parameter can be used to control the recall/precision trade-off, since increasing it adds more position-pairs to the alignment, thus increasing recall, while decreasing it increases precision.

Figures 4 compares the performance of the CRF posterior-decoding MCAs with the baseline MCAs. The different MCAs were produced by varying the gap and match factors in the case of the posterior-decoding, and δ for the baseline MCAs. The CRF curve clearly dominates the baseline curve. However, they do overlap in range between 0.52 and 0.55 recall (0.84 and 0.90 precision). This is prob-

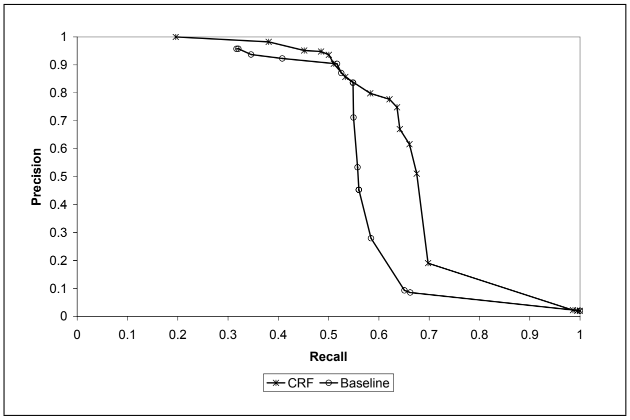


Figure 4: **Recall/Precision curve of MCAs comparing CRF with posterior decoding to normalized-edit-distance baseline.**

ably a range in which for this particular MCA the orthographic similarity is the most dominant feature. While the baseline curve drops sharply after that range, the posterior-decoding curve keeps improving recall up to 0.636 at 0.748 precision, before a major drop in precision. The additional recall is due to the ability of the CRF model to incorporate multiple overlapping features. In particular, the domain-specific features are important for aligning words and phrases that have little or no orthographic similarity. At the other end of the overlap range, the posterior-decoding achieves better precision than the baseline for the same recall levels. For example, the posterior decoding gets 0.381 recall at 0.982 precision compared with 0.346 at 0.937 for the baseline.

Unlike the pairwise alignment case, the neutral settings of the gap and match factors did not result in the best AMA score. This is due to the equivalence closure heuristic that results in MCAs that are too dense, since a single link between two equivalence classes causes them to merge. The best AMA score (0.886) is obtained by reducing the gap-factor to 0.5 and match-factor to 0.45, in order to compensate for the effect of the equivalence closure heuristic. For comparison, the best F_1-measure (0.690) is achieved by setting the gap and match factors to 0.75.

An error analysis on the latter MCA shows that out of 1400 unique errors, 1194 (85.3%) are false negatives (FN) and 206 (14.7%) false positives (FP). Most errors (more than 600) are due to misalignment of subtypes (e.g., *cdc*, *cdc6*, *cdc25A*), opposites (e.g., *phosphorylated* and *unphosphorylated*) and complex entities (e.g., *cell cycle* v.s. *cell line*). In addition, a large portion of FN errors are due to not aligning entities that belong to just four equivalence classes (e.g., 97 FN errors caused by terms in the class of *motif*, *site* and *domain*). Other types of errors include not aligning plural and singular forms of the same entities, aligning only part of multi-word entities, and incorrectly aligning orthographically similar entities that belong to different classes.

7 Conclusions

We have shown how to derive a posterior-decoding algorithm that aims at maximizing the expected utility for the MCA problem, as a substitute for the sequence-annealing algorithm for MSA. Adding a gap and match factor to the utility function allows control over the recall/precision trade-off when using posterior-decoding. Another advantage of optimizing the expected utility with posterior-decoding methods is the decoupling from the probabilistic model that generated the posterior probabilities. This allows the use of CRFs instead of HMMs with a similar posterior decoding algorithm.

Our experiments were limited by the size of the labeled data. However, the results support the theoretical predictions, and demonstrate the advantage of posterior-decoding over Viterbi decoding.

Since citances are still a relatively unexplored resource, it is still unclear whether the formulation we presented here for citance alignment is the most useful for applications that use citances for comparative analysis of bioscience text. Unlike biological sequence alignment, citance alignments are much more subjective, as they depend on a loose definition of semantic homology between entities. Even the definition of the basic entities can vary, since in many cases noun-compounds and other multi-word entities seem to be a more natural choice for basic elements of semantic homology and alignment. However, automatic segmentation and entity recognition are still difficult tasks in the bioscience text domain and so new methods are worth investigating.

Acknowledgements: We thank Phil Blunsom and Trevor Cohn for sharing their CRF-based word alignment code. This research was supported in part by NSF DBI 0317510.

References

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 65–72, Sydney, Australia, July. Association for Computational Linguistics.

Josias Braun-Blanquet. 1932. *Plant sociology: the study of plant communities*. McGraw-Hill, New York.

Francis Caillez and Pascale Kuntz. 1996. A contribution to the study of the metric and euclidean structures of dissimilarities. *Psychometrika*, 61(2):241–253.

Pedros Domingos. 2000. A unified bias-variance decomposition and its applications. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 231–238, Stanford, CA. Morgan Kaufmann.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological sequence analysis. Probablistic models of proteins and nucleic acids*. Cambridge University Press.

Jacob Falck, Niels Mailand, Randi G. Syljuasen, Jiri Bartek, and Jiri Lukas. 2001. The ATM-Chk2-Cdc25A checkpoint pathway guards against radioresistant DNA synthesis. *Nature*, 410(6830):842–847.

Eugene Garfield. 1955. Citation indexes for science: A new dimension in documentation through association of ideas. *Science*, 122(3159):108–111.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. Citeseer: an automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98. ACM Press.

Lynette Hirschman, Jong C. Park, Junichi Tsujii, Limsoon Wong, and Cathy H. Wu. 2002. Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18(12):1553–1561.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I. Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 112–119, New York City, USA, June. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA, June. Association for Computational Linguistics.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA.

Ben-Ami Lipetz. 1965. Improvements of the selectivity of citation indexes to science literature through inclusion of citation relationship indicators. *American Documentation*, 16:81–90.

Mengxiong Liu. 1993. Progress in documentation. the complexities of citation practice: A review of citation studies. *Journal of Documentation*, 49(4):370–408.

Evgeny Matusov, Richard Zens, and Hermann Ney. 2004. Symmetric word alignments for statistical machine translation. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 219, Morristown, NJ, USA. Association for Computational Linguistics.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10, Edmonton, Alberta, Canada, May 31. Association for Computational Linguistics.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *HLT/EMNLP*, pages 81–88.

Preslav I. Nakov, Ariel S. Schwartz, and Marti A. Hearst. 2004. Citances: Citation sentences for semantic analysis of bioscience text. In *SIGIR'04 Workshop on Search and Discovery in Bioinformatics*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Gian-Carlo Rota. 1964. The number of partitions of a set. *The American Mathematical Monthly*, 71(5):498–504, may.

Ralf Schlüter, Thomas Scharrenbach, Volker Steinbiss, and Hermann Ney. 2005. Bayes risk minimization using metric loss functions. In *Proceedings of the European Conference on Speech Communication and Technology, Interspeech*, pages 1449–1452, Portugal, September.

Ariel S. Schwartz and Lior Pachter. 2007. Multiple alignment by sequence annealing. *Bioinformatics*, 23(2):e24–29.

Ariel S. Schwartz, Eugene W. Myers, and Lior Pachter. 2006. Alignment metric accuracy. *arXiv:q-bio.QM/0510052*.

Ariel S. Schwartz. 2007. *Posterior Decoding Methods for Optimization and Accuracy Control of Multiple Alignments*. Ph.D. thesis, EECS Department, University of California, Berkeley.

Howard D. White. 2004. Citation analysis and discourse analysis revisited. *Applied Linguistics*, 25(1):89–116.

Large Language Models in Machine Translation

Thorsten Brants   Ashok C. Popat   Peng Xu   Franz J. Och   Jeffrey Dean

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
{brants,popat,xp,och,jeff}@google.com

Abstract

This paper reports on the benefits of large-scale statistical language modeling in machine translation. A distributed infrastructure is proposed which we use to train on up to 2 trillion tokens, resulting in language models having up to 300 billion n-grams. It is capable of providing smoothed probabilities for fast, single-pass decoding. We introduce a new smoothing method, dubbed *Stupid Backoff*, that is inexpensive to train on large data sets and approaches the quality of Kneser-Ney Smoothing as the amount of training data increases.

1 Introduction

Given a source-language (e.g., French) sentence $\mathbf{f}$, the problem of *machine translation* is to automatically produce a target-language (e.g., English) translation $\hat{\mathbf{e}}$. The mathematics of the problem were formalized by (Brown et al., 1993), and re-formulated by (Och and Ney, 2004) in terms of the optimization

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}, \mathbf{f}) \qquad (1)$$

where $\{h_m(\mathbf{e}, \mathbf{f})\}$ is a set of M *feature functions* and $\{\lambda_m\}$ a set of *weights*. One or more feature functions may be of the form $h(\mathbf{e}, \mathbf{f}) = h(\mathbf{e})$, in which case it is referred to as a *language model*.

We focus on n-gram language models, which are trained on unlabeled monolingual text. As a general rule, more data tends to yield better language models. Questions that arise in this context include: (1) How might one build a language model that allows scaling to very large amounts of training data? (2) How much does translation performance improve as the size of the language model increases? (3) Is there a point of diminishing returns in performance as a function of language model size?

This paper proposes one possible answer to the first question, explores the second by providing learning curves in the context of a particular statistical machine translation system, and hints that the third may yet be some time in answering. In particular, it proposes a *distributed* language model training and deployment infrastructure, which allows direct and efficient integration into the hypothesis-search algorithm rather than a follow-on re-scoring phase. While it is generally recognized that two-pass decoding can be very effective in practice, single-pass decoding remains conceptually attractive because it eliminates a source of potential information loss.

2 N-gram Language Models

Traditionally, statistical language models have been designed to assign probabilities to strings of words (or tokens, which may include punctuation, etc.). Let $w_1^L = (w_1, \ldots, w_L)$ denote a string of L tokens over a fixed vocabulary. An *n-gram language model* assigns a probability to w_1^L according to

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_1^{i-1}) \approx \prod_{i=1}^{L} \hat{P}(w_i | w_{i-n+1}^{i-1}) \qquad (2)$$

where the approximation reflects a Markov assumption that only the most recent $n-1$ tokens are relevant when predicting the next word.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 858–867, Prague, June 2007. ©2007 Association for Computational Linguistics

For any substring w_i^j of w_1^L, let $f(w_i^j)$ denote the frequency of occurrence of that substring in another given, fixed, usually very long target-language string called the *training data*. The maximum-likelihood (ML) probability estimates for the n-grams are given by their relative frequencies

$$r(w_i|w_{i-n+1}^{i-1}) = \frac{f(w_{i-n+1}^i)}{f(w_{i-n+1}^{i-1})}. \quad (3)$$

While intuitively appealing, Eq. (3) is problematic because the denominator and / or numerator might be zero, leading to inaccurate or undefined probability estimates. This is termed the *sparse data* problem. For this reason, the ML estimate must be modified for use in practice; see (Goodman, 2001) for a discussion of n-gram models and smoothing.

In principle, the predictive accuracy of the language model can be improved by increasing the order of the n-gram. However, doing so further exacerbates the sparse data problem. The present work addresses the challenges of processing an amount of training data sufficient for higher-order n-gram models and of storing and managing the resulting values for efficient use by the decoder.

3 Related Work on Distributed Language Models

The topic of large, distributed language models is relatively new. Recently a two-pass approach has been proposed (Zhang et al., 2006), wherein a lower-order n-gram is used in a hypothesis-generation phase, then later the K-best of these hypotheses are re-scored using a large-scale distributed language model. The resulting translation performance was shown to improve appreciably over the hypothesis deemed best by the first-stage system. The amount of data used was 3 billion words.

More recently, a large-scale distributed language model has been proposed in the contexts of speech recognition and machine translation (Emami et al., 2007). The underlying architecture is similar to (Zhang et al., 2006). The difference is that they integrate the distributed language model into their machine translation decoder. However, they don't report details of the integration or the efficiency of the approach. The largest amount of data used in the experiments is 4 billion words.

Both approaches differ from ours in that they store corpora in suffix arrays, one sub-corpus per worker, and serve raw counts. This implies that all workers need to be contacted for each n-gram request. In our approach, smoothed probabilities are stored and served, resulting in exactly one worker being contacted per n-gram for simple smoothing techniques, and in exactly two workers for smoothing techniques that require context-dependent backoff. Furthermore, suffix arrays require on the order of 8 bytes per token. Directly storing 5-grams is more efficient (see Section 7.2) and allows applying count cutoffs, further reducing the size of the model.

4 Stupid Backoff

State-of-the-art smoothing uses variations of context-dependent backoff with the following scheme:

$$P(w_i|w_{i-k+1}^{i-1}) =$$
$$\begin{cases} \rho(w_{i-k+1}^i) & \text{if } (w_{i-k+1}^i) \text{ is found} \\ \lambda(w_{i-k+1}^{i-1})P(w_{i-k+2}^i) & \text{otherwise} \end{cases} \quad (4)$$

where $\rho(\cdot)$ are pre-computed and stored probabilities, and $\lambda(\cdot)$ are back-off weights. As examples, Kneser-Ney Smoothing (Kneser and Ney, 1995), Katz Backoff (Katz, 1987) and linear interpolation (Jelinek and Mercer, 1980) can be expressed in this scheme (Chen and Goodman, 1998). The recursion ends at either unigrams or at the uniform distribution for zero-grams.

We introduce a similar but simpler scheme, named *Stupid Backoff*[1], that does not generate normalized probabilities. The main difference is that we don't apply any discounting and instead directly use the relative frequencies (S is used instead of P to emphasize that these are not probabilities but scores):

$$S(w_i|w_{i-k+1}^{i-1}) =$$
$$\begin{cases} \dfrac{f(w_{i-k+1}^i)}{f(w_{i-k+1}^{i-1})} & \text{if } f(w_{i-k+1}^i) > 0 \\ \alpha S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases} \quad (5)$$

[1] The name originated at a time when we thought that such a simple scheme cannot possibly be good. Our view of the scheme changed, but the name stuck.

In general, the backoff factor α may be made to depend on k. Here, a single value is used and heuristically set to $\alpha = 0.4$ in all our experiments[2]. The recursion ends at unigrams:

$$S(w_i) = \frac{f(w_i)}{N} \tag{6}$$

with N being the size of the training corpus.

Stupid Backoff is inexpensive to calculate in a distributed environment while approaching the quality of Kneser-Ney smoothing for large amounts of data. The lack of normalization in Eq. (5) does not affect the functioning of the language model in the present setting, as Eq. (1) depends on relative rather than absolute feature-function values.

5 Distributed Training

We use the *MapReduce* programming model (Dean and Ghemawat, 2004) to train on terabytes of data and to generate terabytes of language models. In this programming model, a user-specified *map* function processes an input key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function aggregates all intermediate values associated with the same key. Typically, multiple *map* tasks operate independently on different machines and on different parts of the input data. Similarly, multiple *reduce* tasks operate independently on a fraction of the intermediate data, which is partitioned according to the intermediate keys to ensure that the same reducer sees all values for a given key. For additional details, such as communication among machines, data structures and application examples, the reader is referred to (Dean and Ghemawat, 2004).

Our system generates language models in three main steps, as described in the following sections.

5.1 Vocabulary Generation

Vocabulary generation determines a mapping of terms to integer IDs, so n-grams can be stored using IDs. This allows better compression than the original terms. We assign IDs according to term frequency, with frequent terms receiving small IDs for efficient variable-length encoding. All words that occur less often than a pre-determined threshold are mapped to a special id marking the unknown word.

The vocabulary generation *map* function reads training text as input. Keys are irrelevant; values are text. It emits intermediate data where keys are terms and values are their counts in the current section of the text. A *sharding* function determines which shard (chunk of data in the *MapReduce* framework) the pair is sent to. This ensures that all pairs with the same key are sent to the same shard. The *reduce* function receives all pairs that share the same key and sums up the counts. Simplified, the *map*, *sharding* and *reduce* functions do the following:

```
Map(string key, string value) {
  // key=docid, ignored; value=document
  array words = Tokenize(value);
  hash_map<string, int> histo;
  for i = 1 .. #words
    histo[words[i]]++;
  for iter in histo
    Emit(iter.first, iter.second);
}

int ShardForKey(string key, int nshards) {
  return Hash(key) % nshards;
}

Reduce(string key, iterator values) {
  // key=term; values=counts
  int sum = 0;
  for each v in values
    sum += ParseInt(v);
  Emit(AsString(sum));
}
```

Note that the *Reduce* function emits only the aggregated value. The output key is the same as the intermediate key and automatically written by *MapReduce*. The computation of counts in the *map* function is a minor optimization over the alternative of simply emitting a count of one for each tokenized word in the array. Figure 1 shows an example for 3 input documents and 2 reduce shards. Which reducer a particular term is sent to is determined by a hash function, indicated by text color. The exact partitioning of the keys is irrelevant; important is that all pairs with the same key are sent to the same reducer.

5.2 Generation of n-Grams

The process of n-gram generation is similar to vocabulary generation. The main differences are that now words are converted to IDs, and we emit n-grams up to some maximum order instead of single

[2] The value of 0.4 was chosen empirically based on good results in earlier experiments. Using multiple values depending on the n-gram order slightly improves results.

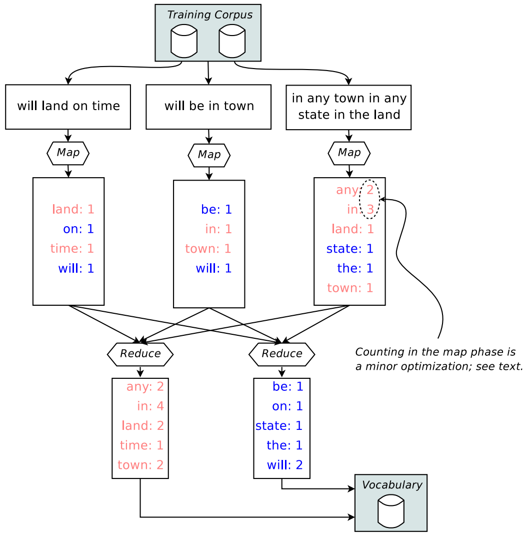


Figure 1: Distributed vocabulary generation.

words. A simplified *map* function does the following:

```
Map(string key, string value) {
  // key=docid, ignored; value=document
  array ids = ToIds(Tokenize(value));
  for i = 1 .. #ids
    for j = 0 .. maxorder-1
      Emit(ids[i-j .. i], "1");
}
```

Again, one may optimize the *Map* function by first aggregating counts over some section of the data and then emit the aggregated counts instead of emitting "1" each time an n-gram is encountered.

The reduce function is the same as for vocabulary generation. The subsequent step of language model generation will calculate relative frequencies $r(w_i|w_{i-k+1}^{i-1})$ (see Eq. 3). In order to make that step efficient we use a sharding function that places the values needed for the numerator and denominator into the same shard.

Computing a hash function on just the first words of n-grams achieves this goal. The required n-grams w_{i-n+1}^{i} and w_{i-n+1}^{i-1} always share the same first word w_{i-n+1}, except for unigrams. For that we need to communicate the total count N to all shards.

Unfortunately, sharding based on the first word only may make the shards very imbalanced. Some terms can be found at the beginning of a huge number of n-grams, e.g. stopwords, some punctuation marks, or the beginning-of-sentence marker. As an example, the shard receiving n-grams starting with the beginning-of-sentence marker tends to be several times the average size. Making the shards evenly sized is desirable because the total runtime of the process is determined by the largest shard.

The shards are made more balanced by hashing based on the first *two* words:

```
int ShardForKey(string key, int nshards) {
  string prefix = FirstTwoWords(key);
  return Hash(prefix) % nshards;
}
```

This requires redundantly storing unigram counts in all shards in order to be able to calculate relative frequencies within shards. That is a relatively small amount of information (a few million entries, compared to up to hundreds of billions of n-grams).

5.3 Language Model Generation

The input to the language model generation step is the output of the n-gram generation step: n-grams and their counts. All information necessary to calculate relative frequencies is available within individual shards because of the sharding function. That is everything we need to generate models with Stupid Backoff. More complex smoothing methods require additional steps (see below).

Backoff operations are needed when the full n-gram is not found. If $r(w_i|w_{i-n+1}^{i-1})$ is not found, then we will successively look for $r(w_i|w_{i-n+2}^{i-1})$, $r(w_i|w_{i-n+3}^{i-1})$, etc. The language model generation step shards n-grams on their last two words (with unigrams duplicated), so all backoff operations can be done within the same shard (note that the required n-grams all share the same last word w_i).

5.4 Other Smoothing Methods

State-of-the-art techniques like Kneser-Ney Smoothing or Katz Backoff require additional, more expensive steps. At runtime, the client needs to additionally request up to 4 backoff factors for each 5-gram requested from the servers, thereby multiplying network traffic. We are not aware of a method that always stores the history backoff factors on the same shard as the longer n-gram without duplicating a large fraction of the entries. This means one needs to contact two shards per n-gram instead of just one for Stupid Backoff. Training requires additional iterations over the data.

	Step 0 context counting	Step 1 unsmoothed probs and interpol. weights	Step 2 interpolated probabilities
Input key	w_{i-n+1}^{i}	(same as Step 0 output)	(same as Step 1 output)
Input value	$f(w_{i-n+1}^{i})$	(same as Step 0 output)	(same as Step 1 output)
Intermediate key	w_{i-n+1}^{i}	w_{i-n+1}^{i-1}	w_{i}^{i-n+1}
Sharding	w_{i-n+1}^{i}	w_{i-n+1}^{i-1}	w_{i-n+1}^{i-n+2}, unigrams duplicated
Intermediate value	$f_{KN}(w_{i-n+1}^{i})$	w_i,$f_{KN}(w_{i-n+1}^{i})$	$\frac{f_{KN}(w_{i-n+1}^{i})-D}{f_{KN}(w_{i-n+1}^{i-1})}$,$\lambda(w_{i-n+1}^{i-1})$
Output value	$f_{KN}(w_{i-n+1}^{i})$	w_i,$\frac{f_{KN}(w_{i-n+1}^{i})-D}{f_{KN}(w_{i-n+1}^{i-1})}$,$\lambda(w_{i-n+1}^{i-1})$	$P_{KN}(w_i\|w_{i-n+1}^{i-1})$, $\lambda(w_{i-n+1}^{i-1})$

Table 1: Extra steps needed for training Interpolated Kneser-Ney Smoothing

Kneser-Ney Smoothing counts lower-order n-grams differently. Instead of the frequency of the $(n-1)$-gram, it uses the number of unique single word contexts the $(n-1)$-gram appears in. We use $f_{KN}(\cdot)$ to jointly denote original frequencies for the highest order and context counts for lower orders. After the n-gram counting step, we process the n-grams again to produce these quantities. This can be done similarly to the n-gram counting using a *MapReduce* (Step 0 in Table 1).

The most commonly used variant of Kneser-Ney smoothing is interpolated Kneser-Ney smoothing, defined recursively as (Chen and Goodman, 1998):

$$\begin{aligned} P_{KN}(w_i|w_{i-n+1}^{i-1}) &= \frac{\max(f_{KN}(w_{i-n+1}^{i})-D,0)}{f_{KN}(w_{i-n+1}^{i-1})} \\ &\quad + \lambda(w_{i-n+1}^{i-1})P_{KN}(w_i|w_{i-n+2}^{i-1}), \end{aligned}$$

where D is a discount constant and $\{\lambda(w_{i-n+1}^{i-1})\}$ are interpolation weights that ensure probabilities sum to one. Two additional major MapReduces are required to compute these values efficiently. Table 1 describes their input, intermediate and output keys and values. Note that output keys are always the same as intermediate keys.

The *map* function of MapReduce 1 emits n-gram histories as intermediate keys, so the *reduce* function gets all n-grams with the same history at the same time, generating unsmoothed probabilities and interpolation weights. MapReduce 2 computes the interpolation. Its *map* function emits reversed n-grams as intermediate keys (hence we use w_i^{i-n+1} in the table). All unigrams are duplicated in every reduce shard. Because the *reducer* function receives intermediate keys in sorted order it can compute smoothed probabilities for all n-gram orders with simple book-keeping.

Katz Backoff requires similar additional steps. The largest models reported here with Kneser-Ney Smoothing were trained on 31 billion tokens. For Stupid Backoff, we were able to use more than 60 times of that amount.

6 Distributed Application

Our goal is to use distributed language models integrated into the first pass of a decoder. This may yield better results than n-best list or lattice rescoring (Ney and Ortmanns, 1999). Doing that for language models that reside in the same machine as the decoder is straight-forward. The decoder accesses n-grams whenever necessary. This is inefficient in a distributed system because network latency causes a constant overhead on the order of milliseconds. Onboard memory is around 10,000 times faster.

We therefore implemented a new decoder architecture. The decoder first queues some number of requests, e.g. 1,000 or 10,000 n-grams, and then sends them together to the servers, thereby exploiting the fact that network requests with large numbers of n-grams take roughly the same time to complete as requests with single n-grams.

The n-best search of our machine translation decoder proceeds as follows. It maintains a graph of the search space up to some point. It then extends each hypothesis by advancing one word position in the source language, resulting in a candidate extension of the hypothesis of zero, one, or more additional target-language words (accounting for the fact that variable-length source-language fragments can correspond to variable-length target-language fragments). In a traditional setting with a local language model, the decoder immediately obtains the necessary probabilities and then (together with scores

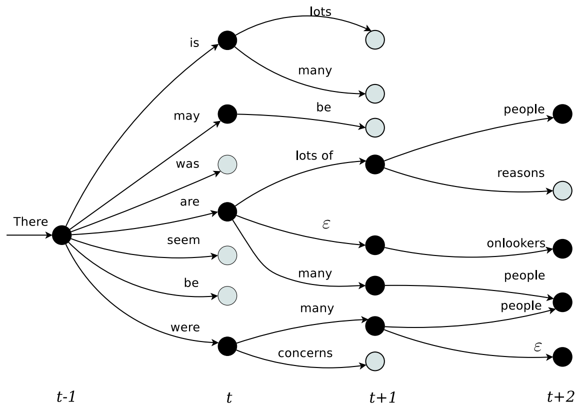


Figure 2: Illustration of decoder graph and batch-querying of the language model.

from other features) decides which hypotheses to keep in the search graph. When using a distributed language model, the decoder first tentatively extends all current hypotheses, taking note of which n-grams are required to score them. These are queued up for transmission as a batch request. When the scores are returned, the decoder re-visits all of these tentative hypotheses, assigns scores, and re-prunes the search graph. It is then ready for the next round of extensions, again involving queuing the n-grams, waiting for the servers, and pruning.

The process is illustrated in Figure 2 assuming a trigram model and a decoder policy of pruning to the four most promising hypotheses. The four active hypotheses (indicated by black disks) at time t are: *There is, There may, There are,* and *There were.* The decoder extends these to form eight new nodes at time $t+1$. Note that one of the arcs is labeled ϵ, indicating that no target-language word was generated when the source-language word was consumed. The n-grams necessary to score these eight hypotheses are *There is lots, There is many, There may be, There are lots, are lots of,* etc. These are queued up and their language-model scores requested in a batch manner. After scoring, the decoder prunes this set as indicated by the four black disks at time $t+1$, then extends these to form five new nodes (one is shared) at time $t+2$. The n-grams necessary to score these hypotheses are *lots of people, lots of reasons, There are onlookers,* etc. Again, these are sent to the server together, and again after scoring the graph is pruned to four active (most promising) hypotheses.

The alternating processes of queuing, waiting and scoring/pruning are done once per word position in a source sentence. The average sentence length in our test data is 22 words (see section 7.1), thus we have 23 rounds[3] per sentence on average. The number of n-grams requested per sentence depends on the decoder settings for beam size, re-ordering window, etc. As an example for larger runs reported in the experiments section, we typically request around 150,000 n-grams per sentence. The average network latency per batch is 35 milliseconds, yielding a total latency of 0.8 seconds caused by the distributed language model for an average sentence of 22 words. If a slight reduction in translation quality is allowed, then the average network latency per batch can be brought down to 7 milliseconds by reducing the number of n-grams requested per sentence to around 10,000. As a result, our system can efficiently use the large distributed language model at decoding time. There is no need for a second pass nor for n-best list rescoring.

We focused on machine translation when describing the queued language model access. However, it is general enough that it may also be applicable to speech decoders and optical character recognition systems.

7 Experiments

We trained 5-gram language models on amounts of text varying from 13 million to 2 trillion tokens. The data is divided into four sets; language models are trained for each set separately[4]. For each training data size, we report the size of the resulting language model, the fraction of 5-grams from the test data that is present in the language model, and the BLEU score (Papineni et al., 2002) obtained by the machine translation system. For smaller training sizes, we have also computed test-set perplexity using Kneser-Ney Smoothing, and report it for comparison.

7.1 Data Sets

We compiled four language model training data sets, listed in order of increasing size:

[3] One additional round for the sentence end marker.

[4] Experience has shown that using multiple, separately trained language models as feature functions in Eq (1) yields better results than using a single model trained on all data.

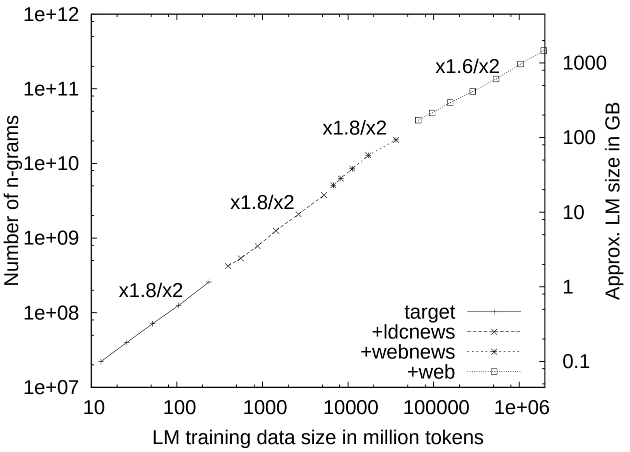


Figure 3: Number of n-grams (sum of unigrams to 5-grams) for varying amounts of training data.

target: The English side of Arabic-English parallel data provided by LDC[5] (237 million tokens).
ldcnews: This is a concatenation of several English news data sets provided by LDC[6] (5 billion tokens).
webnews: Data collected over several years, up to December 2005, from web pages containing predominantly English news articles (31 billion tokens).
web: General web data, which was collected in January 2006 (2 trillion tokens).

For testing we use the "NIST" part of the 2006 Arabic-English NIST MT evaluation set, which is not included in the training data listed above[7]. It consists of 1797 sentences of newswire, broadcast news and newsgroup texts with 4 reference translations each. The test set is used to calculate translation BLEU scores. The English side of the set is also used to calculate perplexities and n-gram coverage.

7.2 Size of the Language Models

We measure the size of language models in total number of n-grams, summed over all orders from 1 to 5. There is no frequency cutoff on the n-grams. There is, however, a frequency cutoff on the vocabulary. The minimum frequency for a term to be included in the vocabulary is 2 for the *target*, *ldcnews* and *webnews* data sets, and 200 for the *web* data set. All terms below the threshold are mapped to a special term `UNK`, representing the unknown word.

	target	***webnews***	***web***
# tokens	237M	31G	1.8T
vocab size	200k	5M	16M
# n-grams	257M	21G	300G
LM size (SB)	2G	89G	1.8T
time (SB)	20 min	8 hours	1 day
time (KN)	2.5 hours	2 days	–
# machines	100	400	1500

Table 2: Sizes and approximate training times for 3 language models with Stupid Backoff (SB) and Kneser-Ney Smoothing (KN).

Figure 3 shows the number of n-grams for language models trained on 13 million to 2 trillion tokens. Both axes are on a logarithmic scale. The right scale shows the approximate size of the served language models in gigabytes. The numbers above the lines indicate the relative increase in language model size: x1.8/x2 means that the number of n-grams grows by a factor of 1.8 each time we double the amount of training data. The values are similar across all data sets and data sizes, ranging from 1.6 to 1.8. The plots are very close to straight lines in the log/log space; linear least-squares regression finds $r^2 > 0.99$ for all four data sets.

The *web* data set has the smallest relative increase. This can be at least partially explained by the higher vocabulary cutoff. The largest language model generated contains approx. 300 billion n-grams.

Table 2 shows sizes and approximate training times when training on the full *target*, *webnews*, and *web* data sets. The processes run on standard current hardware with the Linux operating system. Generating models with Kneser-Ney Smoothing takes 6 – 7 times longer than generating models with Stupid Backoff. We deemed generation of Kneser-Ney models on the *web* data as too expensive and therefore excluded it from our experiments. The estimated runtime for that is approximately one week on 1500 machines.

[5] `http://www.nist.gov/speech/tests/mt/doc/LDCLicense-mt06.pdf` contains a list of parallel resources provided by LDC.

[6] The bigger sets included are LDC2005T12 (Gigaword, 2.5B tokens), LDC93T3A (Tipster, 500M tokens) and LDC2002T31 (Acquaint, 400M tokens), plus many smaller sets.

[7] The test data was generated after 1-Feb-2006; all training data was generated before that date.

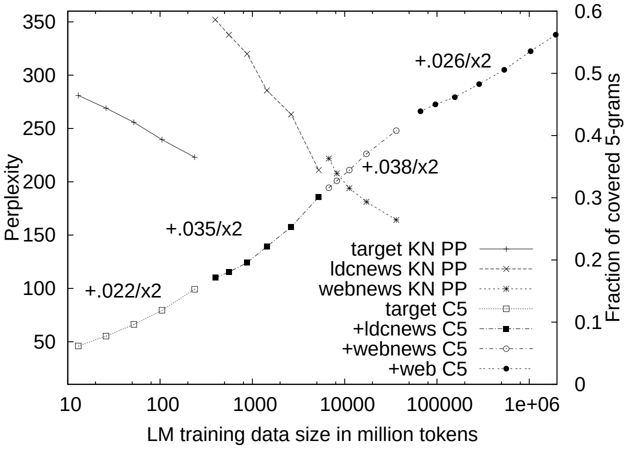


Figure 4: Perplexities with Kneser-Ney Smoothing (KN PP) and fraction of covered 5-grams (C5).

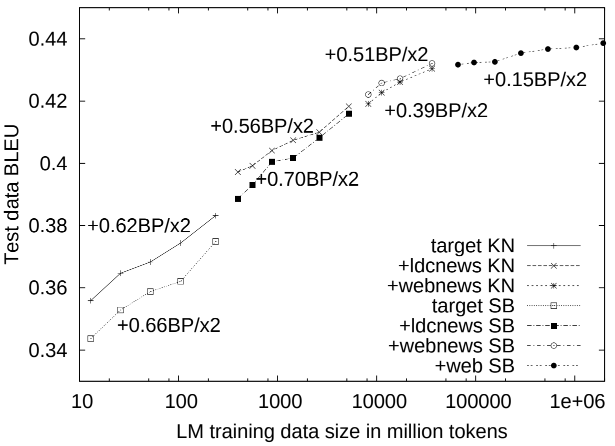


Figure 5: BLEU scores for varying amounts of data using Kneser-Ney (KN) and Stupid Backoff (SB).

7.3 Perplexity and n-Gram Coverage

A standard measure for language model quality is perplexity. It is measured on test data $T = w_1^{|T|}$:

$$PP(T) = e^{-\frac{1}{|T|}\sum\limits_{i=1}^{|T|} \log p(w_i|w_{i-n+1}^{i-1})} \tag{7}$$

This is the inverse of the average conditional probability of a next word; lower perplexities are better. Figure 4 shows perplexities for models with Kneser-Ney smoothing. Values range from 280.96 for 13 million to 222.98 for 237 million tokens *target* data and drop nearly linearly with data size ($r^2 = 0.998$). Perplexities for *ldcnews* range from 351.97 to 210.93 and are also close to linear ($r^2 = 0.987$), while those for *webnews* data range from 221.85 to 164.15 and flatten out near the end. Perplexities are generally high and may be explained by the mixture of genres in the test data (newswire, broadcast news, newsgroups) while our training data is predominantly written news articles. Other held-out sets consisting predominantly of newswire texts receive lower perplexities by the same language models, e.g., using the full *ldcnews* model we find perplexities of 143.91 for the NIST MT 2005 evaluation set, and 149.95 for the NIST MT 2004 set.

Note that the perplexities of the different language models are not directly comparable because they use different vocabularies. We used a fixed frequency cutoff, which leads to larger vocabularies as the training data grows. Perplexities tend to be higher with larger vocabularies.

Perplexities cannot be calculated for language models with Stupid Backoff because their scores are not normalized probabilities. In order to nevertheless get an indication of potential quality improvements with increased training sizes we looked at the 5-gram coverage instead. This is the fraction of 5-grams in the test data set that can be found in the language model training data. A higher coverage will result in a better language model if (as we hypothesize) estimates for seen events tend to be better than estimates for unseen events. This fraction grows from 0.06 for 13 million tokens to 0.56 for 2 trillion tokens, meaning 56% of all 5-grams in the test data are known to the language model.

Increase in coverage depends on the training data set. Within each set, we observe an almost constant growth (correlation $r^2 \geq 0.989$ for all sets) with each doubling of the training data as indicated by numbers next to the lines. The fastest growth occurs for *webnews* data (+0.038 for each doubling), the slowest growth for *target* data (+0.022/x2).

7.4 Machine Translation Results

We use a state-of-the-art machine translation system for translating from Arabic to English that achieved a competitive BLEU score of 0.4535 on the Arabic-English NIST subset in the 2006 NIST machine translation evaluation[8]. Beam size and re-ordering window were reduced in order to facilitate a large

[8] See `http://www.nist.gov/speech/tests/mt/mt06eval_official_results.html` for more results.

number of experiments. Additionally, our NIST evaluation system used a mixture of 5, 6, and 7-gram models with optimized stupid backoff factors for each order, while the learning curve presented here uses a fixed order of 5 and a single fixed backoff factor. Together, these modifications reduce the BLEU score by 1.49 BLEU points (BP)[9] at the largest training size. We then varied the amount of language model training data from 13 million to 2 trillion tokens. All other parts of the system are kept the same.

Results are shown in Figure 5. The first part of the curve uses *target* data for training the language model. With Kneser-Ney smoothing (KN), the BLEU score improves from 0.3559 for 13 million tokens to 0.3832 for 237 million tokens. At such data sizes, Stupid Backoff (SB) with a constant backoff parameter $\alpha = 0.4$ is around 1 BP worse than KN. On average, one gains 0.62 BP for each doubling of the training data with KN, and 0.66 BP per doubling with SB. Differences of more than 0.51 BP are statistically significant at the 0.05 level using bootstrap resampling (Noreen, 1989; Koehn, 2004).

We then add a second language model using *ldcnews* data. The first point for *ldcnews* shows a large improvement of around 1.4 BP over the last point for *target* for both KN and SB, which is approximately twice the improvement expected from doubling the amount of data. This seems to be caused by adding a new domain and combining two models. After that, we find an improvement of 0.56–0.70 BP for each doubling of the *ldcnews* data. The gap between Kneser-Ney Smoothing and Stupid Backoff narrows, starting with a difference of 0.85 BP and ending with a not significant difference of 0.24 BP.

Adding a third language models based on *webnews* data does not show a jump at the start of the curve. We see, however, steady increases of 0.39–0.51 BP per doubling. The gap between Kneser-Ney and Stupid Backoff is gone, all results with Stupid Backoff are actually *better* than Kneser-Ney, but the differences are not significant.

We then add a fourth language model based on *web* data and Stupid Backoff. Generating Kneser-Ney models for these data sizes is extremely expensive and is therefore omitted. The fourth model shows a small but steady increase of 0.15 BP per doubling, surpassing the best Kneser-Ney model (trained on less data) by 0.82 BP at the largest size. Goodman (2001) observed that Kneser-Ney Smoothing dominates other schemes over a broad range of conditions. Our experiments confirm this advantage at smaller language model sizes, but show the advantage disappears at larger data sizes.

The amount of benefit from doubling the training size is partly determined by the domains of the data sets[10]. The improvements are almost linear on the log scale within the sets. Linear least-squares regression shows correlations $r^2 > 0.96$ for all sets and both smoothing methods, thus we expect to see similar improvements when further increasing the sizes.

8 Conclusion

A distributed infrastructure has been described to train and apply large-scale language models to machine translation. Experimental results were presented showing the effect of increasing the amount of training data to up to 2 trillion tokens, resulting in a 5-gram language model size of up to 300 billion n-grams. This represents a gain of about two orders of magnitude in the amount of training data that can be handled over that reported previously in the literature (or three-to-four orders of magnitude, if one considers only single-pass decoding). The infrastructure is capable of scaling to larger amounts of training data and higher n-gram orders.

The technique is made efficient by judicious batching of score requests by the decoder in a server-client architecture. A new, simple smoothing technique well-suited to distributed computation was proposed, and shown to perform as well as more sophisticated methods as the size of the language model increases.

Significantly, we found that translation quality as indicated by BLEU score continues to improve with increasing language model size, at even the largest sizes considered. This finding underscores the value of being able to train and apply very large language models, and suggests that further performance gains may be had by pursuing this direction further.

[9] 1 BP = 0.01 BLEU. We show system scores as BLEU, differences as BP.

[10] There is also an effect of the order in which we add the models. As an example, *web* data yields +0.43 BP/x2 when added as the second model. A discussion of this effect is omitted due to space limitations.

References

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard, Cambridge, MA, USA.

Jeffrey Dean and Sanjay Ghemawat. 2004. Mapreduce: Simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementation (OSDI-04)*, San Francisco, CA, USA.

Ahmad Emami, Kishore Papineni, and Jeffrey Sorensen. 2007. Large-scale distributed language modeling. In *Proceedings of ICASSP-2007*, Honolulu, HI, USA.

Joshua Goodman. 2001. A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research, Redmond, WA, USA.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*, pages 381–397. North Holland.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3).

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP-04*, Barcelona, Spain.

Hermann Ney and Stefan Ortmanns. 1999. Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, 16(5):64–83.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL-02*, pages 311–318, Philadelphia, PA, USA.

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. 2006. Distributed language modeling for n-best list re-ranking. In *Proceedings of EMNLP-2006*, pages 216–223, Sydney, Australia.

Factored Translation Models

Philipp Koehn and Hieu Hoang
pkoehn@inf.ed.ac.uk, H.Hoang@sms.ed.ac.uk
School of Informatics
University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW
Scotland, United Kingdom

Abstract

We present an extension of phrase-based statistical machine translation models that enables the straight-forward integration of additional annotation at the word-level — may it be linguistic markup or automatically generated word classes. In a number of experiments we show that factored translation models lead to better translation performance, both in terms of automatic scores, as well as more grammatical coherence.

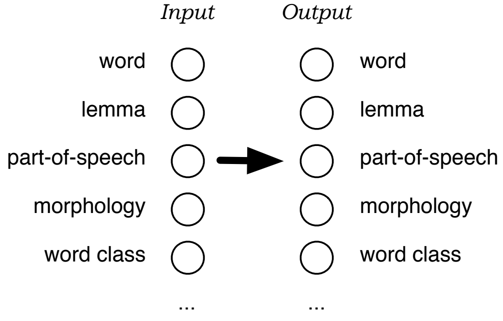


Figure 1: Factored representations of input and output words incorporate additional annotation into the statistical translation model.

1 Introduction

The current state-of-the-art approach to statistical machine translation, so-called phrase-based models, is limited to the mapping of small text chunks without any explicit use of linguistic information, may it be morphological, syntactic, or semantic. Such additional information has been demonstrated to be valuable by integrating it in pre-processing or post-processing steps.

However, a tighter integration of linguistic information into the translation model is desirable for two reasons:

- Translation models that operate on more general representations, such as lemmas instead of surface forms of words, can draw on richer statistics and overcome the data sparseness problems caused by limited training data.
- Many aspects of translation can be best explained on a morphological, syntactic, or semantic level. Having such information available to the translation model allows the direct modeling of these aspects. For instance: reordering at the sentence level is mostly driven by general syntactic principles, local agreement constraints show up in morphology, etc.

Therefore, we extended the phrase-based approach to statistical translation to tightly integrate additional information. The new approach allows additional annotation at the word level. A word in our framework is not only a token, but a vector of factors that represent different levels of annotation (see Figure 1).

We report on experiments with factors such as surface form, lemma, part-of-speech, morphological features such as gender, count and case, automatic word classes, true case forms of words, shallow syntactic tags, as well as dedicated factors to ensure agreement between syntactically related items.

This paper describes the motivation, the modeling aspects and the computationally efficient decoding methods of factored translation models. We present briefly results for a number of language pairs. However, the focus of this paper is the description of the approach. Detailed experimental results will be described in forthcoming papers.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 868–876, Prague, June 2007. ©2007 Association for Computational Linguistics

2 Related Work

Many attempts have been made to add richer information to statistical machine translation models. Most of these focus on the pre-processing of the input to the statistical system, or the post-processing of its output. Our framework is more general and goes beyond recent work on models that back off to representations with richer statistics (Nießen and Ney, 2001; Yang and Kirchhoff, 2006; Talbot and Osborne, 2006) by keeping a more complex representation throughout the translation process.

Rich morphology often poses a challenge to statistical machine translation, since a multitude of word forms derived from the same lemma fragment the data and lead to sparse data problems. If the input language is morphologically richer than the output language, it helps to stem or segment the input in a pre-processing step, before passing it on to the translation system (Lee, 2004; Sadat and Habash, 2006).

Structural problems have also been addressed by pre-processing: Collins et al. (2005) reorder the input to a statistical system to closer match the word order of the output language.

On the other end of the translation pipeline, additional information has been used in post-processing. Och et al. (2004) report minor improvements with linguistic features on a Chinese-English task, Koehn and Knight (2003) show some success in re-ranking noun phrases for German-English. In their approaches, first, an n-best list with the best translations is generated for each input sentence. Then, the n-best list is enriched with additional features, for instance by syntactically parsing each candidate translation and adding a parse score. The additional features are used to rescore the n-best list, resulting possibly in a better best translation for the sentence.

The goal of integrating syntactic information into the translation model has prompted many researchers to pursue tree-based transfer models (Wu, 1997; Alshawi et al., 1998; Yamada and Knight, 2001; Melamed, 2004; Menezes and Quirk, 2005; Galley et al., 2006), with increasingly encouraging results. Our goal is complementary to these efforts: we are less interested in recursive syntactic structure, but in richer annotation at the word level. In future work, these approaches may be combined.

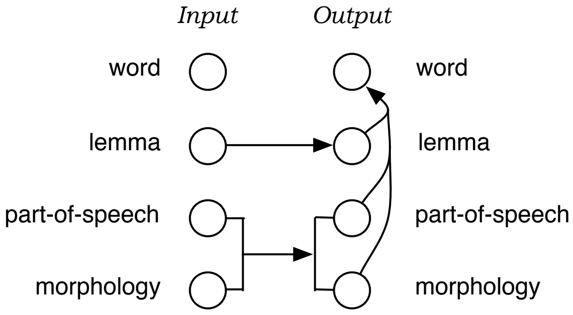


Figure 2: Example factored model: morphological analysis and generation, decomposed into three mapping steps (translation of lemmas, translation of part-of-speech and morphological information, generation of surface forms).

3 Motivating Example: Morphology

One example to illustrate the short-comings of the traditional surface word approach in statistical machine translation is the poor handling of morphology. Each word form is treated as a token in itself. This means that the translation model treats, say, the word *house* completely independent of the word *houses*. Any instance of *house* in the training data does not add any knowledge to the translation of *houses*.

In the extreme case, while the translation of *house* may be known to the model, the word *houses* may be unknown and the system will not be able to translate it. While this problem does not show up as strongly in English — due to the very limited morphological inflection in English — it does constitute a significant problem for morphologically rich languages such as Arabic, German, Czech, etc.

Thus, it may be preferably to model translation between morphologically rich languages on the level of lemmas, and thus pooling the evidence for different word forms that derive from a common lemma. In such a model, we would want to translate lemma and morphological information separately, and combine this information on the output side to ultimately generate the output surface words.

Such a model can be defined straight-forward as a factored translation model. See Figure 2 for an illustration of this model in our framework.

Note that while we illustrate the use of factored translation models on such a linguistically motivated example, our framework also applies to models that incorporate statistically defined word classes, or any other annotation.

4 Decomposition of Factored Translation

The translation of factored representations of input words into the factored representations of output words is broken up into a sequence of **mapping steps** that either **translate** input factors into output factors, or **generate** additional output factors from existing output factors.

Recall the example of a factored model motivated by morphological analysis and generation. In this model the translation process is broken up into the following three mapping steps:

1. **Translate** input lemmas into output lemmas
2. **Translate** morphological and POS factors
3. **Generate** surface forms given the lemma and linguistic factors

Factored translation models build on the phrase-based approach (Koehn et al., 2003) that breaks up the translation of a sentence into the translation of small text chunks (so-called phrases). This approach implicitly defines a segmentation of the input and output sentences into phrases. See an example in Figure 3.

Our current implementation of factored translation models follows strictly the phrase-based approach, with the additional decomposition of phrase translation into a sequence of mapping steps. Translation steps map factors in input phrases to factors in output phrases. Generation steps map output factors within individual output words. To reiterate: all translation steps operate on the phrase level, while all generation steps operate on the word level. Since all mapping steps operate on the same phrase segmentation of the input and output sentence into phrase pairs, we call these **synchronous factored models**.

Let us now take a closer look at one example, the translation of the one-word phrase *häuser* into English. The representation of *häuser* in German is: surface-form *häuser* | lemma *haus* | part-of-speech *NN* | count *plural* | case *nominative* | gender *neutral*.

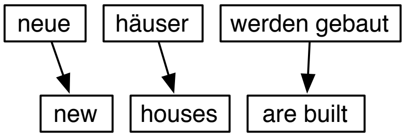


Figure 3: Example sentence translation by a standard phrase model. Factored models extend this approach.

The three mapping steps in our morphological analysis and generation model may provide the following applicable mappings:

1. **Translation:** Mapping lemmas
   - *haus → house, home, building, shell*
2. **Translation:** Mapping morphology
   - *NN|plural-nominative-neutral → NN|plural, NN|singular*
3. **Generation:** Generating surface forms
   - *house|NN|plural → houses*
   - *house|NN|singular → house*
   - *home|NN|plural → homes*
   - ...

We call the application of these mapping steps to an input phrase **expansion**. Given the multiple choices for each step (reflecting the ambiguity in translation), each input phrase may be expanded into a list of translation options. The German *häuser|haus|NN|plural-nominative-neutral* may be expanded as follows:

1. **Translation:** Mapping lemmas
   { *?|house|?|?, ?|home|?|?, ?|building|?|?, ?|shell|?|?* }
2. **Translation:** Mapping morphology
   { *?|house|NN|plural, ?|home|NN|plural, ?|building|NN|plural, ?|shell|NN|plural, ?|house|NN|singular,* ... }
3. **Generation:** Generating surface forms
   { *houses|house|NN|plural, homes|home|NN|plural, buildings|building|NN|plural, shells|shell|NN|plural, house|house|NN|singular,* ... }

5 Statistical Model

Factored translation models follow closely the statistical modeling approach of phrase-based models (in fact, phrase-based models are a special case of factored models). The main difference lies in the preparation of the training data and the type of models learned from the data.

5.1 Training

The training data (a parallel corpus) has to be annotated with the additional factors. For instance, if we want to add part-of-speech information on the input and output side, we need to obtain part-of-speech tagged training data. Typically this involves running automatic tools on the corpus, since manually annotated corpora are rare and expensive to produce.

Next, we need to establish a word-alignment for all the sentences in the parallel training corpus. Here, we use the same methodology as in phrase-based models (typically symmetrized GIZA++ alignments). The word alignment methods may operate on the surface forms of words, or on any of the other factors. In fact, some preliminary experiments have shown that word alignment based on lemmas or stems yields improved alignment quality.

Each mapping step forms a component of the overall model. From a training point of view this means that we need to learn translation and generation tables from the word-aligned parallel corpus and define scoring methods that help us to choose between ambiguous mappings.

Phrase-based translation models are acquired from a word-aligned parallel corpus by extracting all phrase-pairs that are consistent with the word alignment. Given the set of extracted phrase pairs with counts, various **scoring functions** are estimated, such as conditional phrase translation probabilities based on relative frequency estimation or lexical translation probabilities based on the words in the phrases.

In our approach, the models for the translation steps are acquired in the same manner from a word-aligned parallel corpus. For the specified factors in the input and output, phrase mappings are extracted. The set of phrase mappings (now over factored representations) is scored based on relative counts and word-based translation probabilities.

The generation distributions are estimated on the output side only. The word alignment plays no role here. In fact, additional monolingual data may be used. The generation model is learned on a word-for-word basis. For instance, for a generation step that maps surface forms to part-of-speech, a table with entries such as *(fish,NN)* is constructed. One or more scoring functions may be defined over this table, in our experiments we used both conditional probability distributions, e.g., $p(\textit{fish}|\textit{NN})$ and $p(\textit{NN}|\textit{fish})$, obtained by maximum likelihood estimation.

An important component of statistical machine translation is the language model, typically an n-gram model over surface forms of words. In the framework of factored translation models, such sequence models may be defined over any factor, or any set of factors. For factors such as part-of-speech tags, building and using higher order n-gram models (7-gram, 9-gram) is straight-forward.

5.2 Combination of Components

As in phrase-based models, factored translation models can be seen as the combination of several components (language model, reordering model, translation steps, generation steps). These components define one or more feature functions that are combined in a log-linear model:

$$p(\mathbf{e}|\mathbf{f}) = \frac{1}{Z} \exp \sum_{i=1}^{n} \lambda_i h_i(\mathbf{e}, \mathbf{f}) \tag{1}$$

Z is a normalization constant that is ignored in practice. To compute the probability of a translation **e** given an input sentence **f**, we have to evaluate each feature function h_i. For instance, the feature function for a bigram language model component is (m is the number of words e_i in the sentence **e**):

$$\begin{aligned} h_{\text{LM}}(\mathbf{e}, \mathbf{f}) &= p_{\text{LM}}(\mathbf{e}) \\ &= p(e_1)\; p(e_2|e_1)..p(e_m|e_{m-1}) \end{aligned} \tag{2}$$

Let us now consider the feature functions introduced by the translation and generation steps of factored translation models. The translation of the input sentence **f** into the output sentence **e** breaks down to a set of phrase translations $\{(\bar{f}_j, \bar{e}_j)\}$.

For a translation step component, each feature function h_{T} is defined over the phrase pairs $(\bar{f}_j, \bar{e}_j)$

given a scoring function τ:

$$h_{\mathrm{T}}(\mathbf{e},\mathbf{f}) = \sum_j \tau(\bar{f}_j, \bar{e}_j) \tag{3}$$

For a generation step component, each feature function h_{G} given a scoring function γ is defined over the output words e_k only:

$$h_{\mathrm{G}}(\mathbf{e},\mathbf{f}) = \sum_k \gamma(e_k) \tag{4}$$

The feature functions follow from the scoring functions (τ, γ) acquired during the training of translation and generation tables. For instance, recall our earlier example: a scoring function for a generation model component that is a conditional probability distribution between input and output factors, e.g., γ(*fish,NN,singular*) $= p$(*NN*|*fish*).

The feature weights λ_i in the log-linear model are determined using a minimum error rate training method, typically Powell's method (Och, 2003).

5.3 Efficient Decoding

Compared to phrase-based models, the decomposition of phrase translation into several mapping steps creates additional computational complexity. Instead of a simple table look-up to obtain the possible translations for an input phrase, now multiple tables have to be consulted and their content combined.

In phrase-based models it is easy to identify the entries in the phrase table that may be used for a specific input sentence. These are called **translation options**. We usually limit ourselves to the top 20 translation options for each input phrase.

The beam search decoding algorithm starts with an empty hypothesis. Then new hypotheses are generated by using all applicable translation options. These hypotheses are used to generate further hypotheses in the same manner, and so on, until hypotheses are created that cover the full input sentence. The highest scoring complete hypothesis indicates the best translation according to the model.

How do we adapt this algorithm for factored translation models? Since all mapping steps operate on the same phrase segmentation, the **expansions** of these mapping steps can be efficiently pre-computed prior to the heuristic beam search, and stored as translation options. For a given input phrase, all possible translation options are thus computed before decoding (recall the example in Section 4, where we carried out the expansion for one input phrase). This means that the fundamental search algorithm does not change.

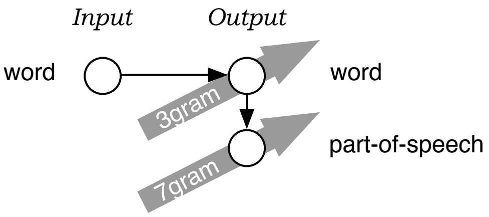


Figure 4: Syntactically enriched output: By generating additional linguistic factors on the output side, high-order sequence models over these factors support syntactical coherence of the output.

However, we need to be careful about combinatorial explosion of the number of translation options given a sequence of mapping steps. In other words, the expansion may create too many translation options to handle. If one or many mapping steps result in a vast increase of (intermediate) expansions, this may be become unmanageable. We currently address this problem by early pruning of expansions, and limiting the number of translation options per input phrase to a maximum number, by default 50. This is, however, not a perfect solution. We are currently working on a more efficient search for the top 50 translation options to replace the current brute-force approach.

6 Experiments

We carried out a number of experiments using the factored translation model framework, incorporating both linguistic information and automatically generated word classes.

This work is implemented as part of the open source Moses[1] system (Koehn et al., 2007). We used the default settings for this system.

6.1 Syntactically Enriched Output

In the first set of experiments, we translate surface forms of words and generate additional output factors from them (see Figure 4 for an illustration). By adding morphological and shallow syntactic infor-

[1] available at `http://www.statmt.org/moses/`

English–German

Model	BLEU
best published result	18.15%
baseline (surface)	18.04%
surface + POS	18.15%
surface + POS + morph	18.22%

English–Spanish

Model	BLEU
baseline (surface)	23.41%
surface + morph	24.66%
surface + POS + morph	24.25%

English–Czech

Model	BLEU
baseline (surface)	25.82%
surface + all morph	27.04%
surface + case/number/gender	27.45%
surface + CNG/verb/prepositions	27.62%

Table 1: Experimental results with syntactically enriched output (part of speech, morphology)

mation, we are able to use high-order sequence models (just like n-gram language models over words) in order to support syntactic coherence of the output. Table 1 summarizes the experimental results.

The English–German systems were trained on the full 751,088 sentence Europarl corpus and evaluated on the WMT 2006 test set (Koehn and Monz, 2006). Adding part-of-speech and morphological factors on the output side and exploiting them with 7-gram sequence models results in minor improvements in BLEU. The model that incorporates both POS and morphology (18.22% BLEU vs. baseline 18.04% BLEU) ensures better local grammatical coherence. The baseline system produces often phrases such as *zur*(to) *zwischenstaatlichen*(inter-governmental) *methoden*(methods), with a mismatch between the determiner (singular) and the noun (plural), while the adjective is ambiguous. In a manual evaluation of intra-NP agreement we found that the factored model reduced the disagreement error within noun phrases of length ≥ 3 from 15% to 4%.

English–Spanish systems were trained on a 40,000 sentence subset of the Europarl corpus. Here, we also used morphological and part-of-speech factors on the output side with an 7-gram sequence model, resulting in absolute improvements of 1.25% (only morph) and 0.84% (morph+POS). Improvements on the full Europarl corpus are smaller.

English-Czech systems were trained on a 20,000 sentence Wall Street Journal corpus. Morphological features were exploited with a 7-gram language model. Experimentation suggests that it is beneficial to carefully consider which morphological features to be used. Adding all features results in lower performance (27.04% BLEU), than considering only case, number and gender (27.45% BLEU) or additionally verbial (person, tense, and aspect) and prepositional (lemma and case) morphology (27.62% BLEU). All these models score well above the baseline of 25.82% BLEU.

An extended description of these experiments is in the JHU workshop report (Koehn et al., 2006).

6.2 Morphological Analysis and Generation

The next model is the one described in our motivating example in Section 4 (see also Figure 2). Instead of translating surface forms of words, we translate word lemma and morphology separately, and generate the surface form of the word on the output side.

We carried out experiments for the language pair German–English, using the 52,185 sentence News Commentary corpus[2]. We report results on the development test set, which is also the out-of-domain test set of the WMT06 workshop shared task (Koehn and Monz, 2006). German morphological analysis and POS tagging was done using LoPar Schmidt and Schulte im Walde (2000), English POS tagging was done with Brill's tagger (Brill, 1995), followed by a simple lemmatizer based on tagging results.

Experimental results are summarized in Table 2. For this data set, we also see an improvement when using a part-of-speech language model — the BLEU score increases from 18.19% to 19.05% — consistent with the results reported in the previous section. However, moving from a surface word translation mapping to a lemma/morphology mapping leads to a deterioration of performance to a BLEU score of 14.46%.

Note that this model completely ignores the surface forms of input words and only relies on the

[2] Made available for the WMT07 workshop shared task `http://www.statmt.org/wmt07/`

German–English

Model	BLEU
baseline (surface)	18.19%
+ POS LM	19.05%
pure lemma/morph model	14.46%
backoff lemma/morph model	19.47%

Table 2: Experimental results with morphological analysis and generation model (Figure 2), using News Commentary corpus

more general lemma and morphology information. While this allows the translation of word forms with known lemma and unknown surface form, on balance it seems to be disadvantage to throw away surface form information.

To overcome this problem, we introduce an alternative path model: Translation options in this model may come either from the surface form model or from the lemma/morphology model we just described. For surface forms with rich evidence in the training data, we prefer surface form mappings, and for surface forms with poor or no evidence in the training data we decompose surface forms into lemma and morphology information and map these separately. The different translation tables form different components in the log-linear model, whose weights are set using standard minimum error rate training methods.

The alternative path model outperforms the surface form model with POS LM, with an BLEU score of 19.47% vs. 19.05%. The test set has 3276 unknown word forms vs 2589 unknown lemmas (out of 26,898 words). Hence, the lemma/morph model is able to translate 687 additional words.

6.3 Use of Automatic Word Classes

Finally, we went beyond linguistically motivated factors and carried out experiments with automatically trained word classes. By clustering words together by their contextual similarity, we are able to find statistically similarities that may lead to more generalized and robust models.

We trained models on the IWSLT 2006 task (39,953 sentences). Compared to a baseline English–Chinese system, adding word classes on the output side as additional factors (in a model as previously illustrated in Figure 4) to be exploited by a 7-gram sequence model, we observe a gain 1.5% BLEU absolute. For more on this experiment, see (Shen et al., 2006).

English–Chinese

Model	BLEU
baseline (surface)	19.54%
surface + word class	21.10%

Table 3: Experimental result with automatic word classes obtained by word clustering

Chinese–English

Recase Method	BLEU
Standard two-pass: SMT + recase	20.65%
Integrated factored model (optimized)	21.08%

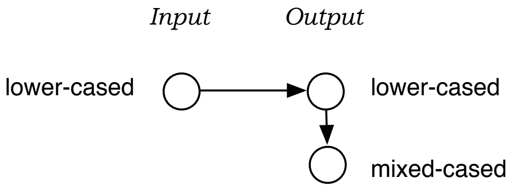


Table 4: Experimental result with integrated recasing (IWSLT 2006 task)

6.4 Integrated Recasing

To demonstrate the versatility of the factored translation model approach, consider the task of recasing (Lita et al., 2003; Wang et al., 2006). Typically in statistical machine translation, the training data is lowercased to generalize over differently cased surface forms — say, *the*, *The*, *THE* — which necessitates a post-processing step to restore case in the output.

With factored translation models, it is possible to integrate this step into the model, by adding a generation step. See Table 4 for an illustration of this model and experimental results on the IWSLT 2006 task (Chinese-English). The integrated recasing model outperform the standard approach with an BLEU score of 21.08% to 20.65%. For more on this experiment, see (Shen et al., 2006).

6.5 Additional Experiments

Factored translation models have also been used for the integration of CCG supertags (Birch et al., 2007), domain adaptation (Koehn and Schroeder, 2007) and for the improvement of English-Czech translation (Bojar, 2007).

7 Conclusion and Future Work

We presented an extension of the state-of-the-art phrase-based approach to statistical machine translation that allows the straight-forward integration of additional information, may it come from linguistic tools or automatically acquired word classes.

We reported on experiments that showed gains over standard phrase-based models, both in terms of automatic scores (gains of up to 2% BLEU), as well as a measure of grammatical coherence. These experiments demonstrate that within the framework of factored translation models additional information can be successfully exploited to overcome some short-comings of the currently dominant phrase-based statistical approach.

The framework of factored translation models is very general. Many more models that incorporate different factors can be quickly built using the existing implementation. We are currently exploring these possibilities, for instance use of syntactic information in reordering and models with augmented input information.

We have not addressed all computational problems of factored translation models. In fact, computational problems hold back experiments with more complex factored models that are theoretically possible but too computationally expensive to carry out. Our current focus is to develop a more efficient implementation that will enable these experiments.

Moreover, we expect to overcome the constraints of the currently implemented *synchronous* factored models by developing a more general *asynchronous* framework, where multiple translation steps may operate on different phrase segmentations (for instance a part-of-speech model for large scale reordering).

Acknowledgments

This work was supported in part under the GALE program of the Defense Advanced Research Projects Agency, Contract No NR0011-06-C-0022 and in part under the EuroMatrix project funded by the European Commission (6th Framework Programme).

We also benefited greatly from a 2006 summer workshop hosted by the Johns Hopkins University and would like thank the other workshop participants for their support and insights, namely Nicola Bertoldi, Ondrej Bojar, Chris Callison-Burch, Alexandra Constantin, Brooke Cowan, Chris Dyer, Marcello Federico, Evan Herbst Christine Moran, Wade Shen, and Richard Zens.

References

Alshawi, H., Bangalore, S., and Douglas, S. (1998). Automatic acquisition of hierarchical transduction models for machine translation. In *Proceedings of the 36th Annual Meeting of the Association of Computational Linguistics (ACL)*.

Birch, A., Osborne, M., and Koehn, P. (2007). CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16, Prague, Czech Republic. Association for Computational Linguistics.

Bojar, O. (2007). English-to-Czech factored machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 232–239, Prague, Czech Republic. Association for Computational Linguistics.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4).

Collins, M., Koehn, P., and Kucerova, I. (2005). Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan. Association for Computational Linguistics.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia. Association for Computational Linguistics.

Koehn, P., Federico, M., Shen, W., Bertoldi, N., Hoang, H., Callison-Burch, C., Cowan, B., Zens, R., Dyer, C., Bojar, O., Moran, C., Constantin, A., and Herbst, E. (2006). Open source toolkit for statistical machine translation: Factored translation models and confusion network decoding. Technical report, John Hopkins University Summer Workshop.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, demonstation session*.

Koehn, P. and Knight, K. (2003). Feature-rich translation of noun phrases. In *41st Annual Meeting of the Association of Computational Linguistics (ACL)*.

Koehn, P. and Monz, C. (2006). Manual and automatic evaluation of machine translation between European languages. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 102–121, New York City. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase based translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Koehn, P. and Schroeder, J. (2007). Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic. Association for Computational Linguistics.

Lee, Y.-S. (2004). Morphological analysis for statistical machine translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Lita, L. V., Ittycheriah, A., Roukos, S., and Kambhatla, N. (2003). tRuEcasIng. In Hinrichs, E. and Roth, D., editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 152–159.

Melamed, I. D. (2004). Statistical machine translation by parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 653–660, Barcelona, Spain.

Menezes, A. and Quirk, C. (2005). Microsoft research treelet translation system: IWSLT evaluation. In *Proc. of the International Workshop on Spoken Language Translation*.

Nießen, S. and Ney, H. (2001). Toward hierarchical models for statistical machine translation of inflected languages. In *Workshop on Data-Driven Machine Translation at 39th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 47–54.

Och, F. J. (2003). Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL)*.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., and Radev, D. (2004). A smorgasbord of features for statistical machine translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Sadat, F. and Habash, N. (2006). Combination of arabic preprocessing schemes for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Schmidt, H. and Schulte im Walde, S. (2000). Robust German noun chunking with a probabilistic context-free grammar. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Shen, W., Zens, R., Bertoldi, N., and Federico, M. (2006). The JHU Workshop 2006 IWSLT System. In *Proc. of the International Workshop on Spoken Language Translation*, pages 59–63, Kyoto, Japan.

Talbot, D. and Osborne, M. (2006). Modelling lexical redundancy for machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 969–976, Sydney, Australia. Association for Computational Linguistics.

Wang, W., Knight, K., and Marcu, D. (2006). Capitalizing machine translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics (ACL)*.

Yang, M. and Kirchhoff, K. (2006). Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Translating Unknown Words by Analogical Learning

Philippe Langlais and **Alexandre Patry**
Dept. I.R.O.
Université de Montréal
{felipe,patryale}@iro.umontreal.ca

Abstract

Unknown words are a well-known hindrance to natural language applications. In particular, they drastically impact machine translation quality. An easy way out commercial translation systems usually offer their users is the possibility to add unknown words and their translations into a dedicated lexicon. Recently, Stroppa and Yvon (2005) have shown how analogical learning alone deals nicely with morphology in different languages. In this study we show that analogical learning offers as well an elegant and effective solution to the problem of identifying potential translations of unknown words.

1 Introduction

Analogical reasoning has received some attention in cognitive science and artificial intelligence (Gentner et al., 2001). It has been for a long time a faculty assessed in the so-called SAT Reasoning tests used in the application process to colleges and universities in the United States. Turney (2006) has shown that it is possible to compute relational similarities in a corpus in order to solve 56% of typical analogical tests quizzed in SAT exams. The interested reader can find in (Lepage, 2003) a particularly dense treatment of analogy, including a fascinating chapter on the history of the notion of analogy.

The concept of *proportional analogy*, denoted $[A : B = C : D]$, is a relation between four entities which reads: "A is to B as C is to D". Among proportional analogies, we distinguish *formal analogies*, that is, ones that arise at the graphical level, such as [*fournit* : *fleurit* = *fournie* : *fleurie*] in French or [*believer* : *unbelievable* = *doer* : *undoable*] in English. Formal analogies are often good indices for deeper analogies (Stroppa and Yvon, 2005).

Lepage and Denoual (2005) presented the system ALEPH, an intriguing example-based system entirely built on top of an automatic formal analogy solver. This system has achieved state-of-the-art performance on the IWSLT task (Eck and Hori, 2005), despite its striking purity. As a matter of fact, ALEPH requires no distances between examples, nor any threshold.[1] It does not even rely on a tokenization device. One reason for its success probably lies in the specificity of the BTEC corpus: short and simple sentences of a narrow domain. It is doubtful that ALEPH would still behave adequately on broader tasks, such as translating news articles.

Stroppa and Yvon (2005) propose a very helpful algebraic description of a formal analogy and describe the theoretical foundations of *analogical learning* which we will recap shortly. They show both its elegance and efficiency on two morphological analysis tasks for three different languages.

Recently, Moreau et al. (2007) showed that formal analogies of a simple kind (those involving suffixation and/or prefixation) offer an effective way to extend queries for improved information retrieval.

In this study, we show that analogical learning can be used as an efficitive method for translating unknown words or phrases. We found that our approach has the potential to propose a valid translation for 80% of *ordinary* unknown words, that is, words that are not proper names, compound words, or numerical expressions. Specific solutions have been proposed for those token types (Chen et al., 1998; Al-Onaizan and Knight, 2002; Koehn and Knight, 2003).

The paper is organized as follows. We first recall

[1] Some heuristics are applied for speeding up the system.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 877–886, Prague, June 2007. ©2007 Association for Computational Linguistics

in Section 2 the principle of analogical learning and describe how it can be applied to the task of enriching a bilingual lexicon. In Section 3, we present the corpora we used in our experiments. We evaluate our approach over two translation tasks in Section 4. We discuss related work in Section 5 and give perspectives of our work in Section 6.

2 Analogical Learning

2.1 Principle

Our approach to bilingual lexical enrichment is an instance of analogical learning described in (Stroppa and Yvon, 2005). A learning set $\mathcal{L} = \{L_1, \ldots, L_N\}$ gathers N observations. A set of features computed on an incomplete observation X defines an input space. The inference task consists in predicting the missing features which belong to an output space. We denote $I(X)$ (resp. $O(X)$) the projection of X into the input (resp. output) space. The inference procedure involves three steps:

1. Building $\mathcal{E}_{\mathcal{I}}(X) = \{(A, B, C) \in \mathcal{L}^3 \,|\, [I(A) : I(B) = I(C) : I(X)]\}$, the set of input *stems*[2] of X, that is the set of triplets (A, B, C) which form with X an analogical equation.
2. Building $\mathcal{E}_{\mathcal{O}}(X) = \{Y \,|\, [O(A) : O(B) = O(C) : Y], \forall (A, B, C) \in \mathcal{E}_{\mathcal{I}}(X)\}$ the set of solutions to the analogical equations obtained by projecting the stems of $\mathcal{E}_{\mathcal{I}}(X)$ into the output space.
3. Selecting $O(X)$ among the elements of $\mathcal{E}_{\mathcal{O}}(X)$.

This inference procedure shares similarities with the K-nearest-neighbor (k-NN) approach. In particular, since no model of the training material is being learned, the training corpus needs to be stored in order to be queried. On the contrary to k-NN, however, the search for closest neighbors does not require any distance, but instead relies on relational similarities. This purity has a cost: while in k-NN inference, neighbors can be found in time linear to the training size, in analogical learning, this operation requires a computation time cubic in N, the number of observations. In many applications of interest, including the one we tackle here, this is simply impractical and heuristics must be applied.

The first and second steps of the inference procedure rely on the existence of an analogical solver, which we sketch in the next section. One important thing to note at this stage, is that an analogical equation may have several solutions, some being legitimate word-forms in a given language, others being not. Thus, it is important to select wisely the generated solutions, therefore Step 3. In practice, the inference procedure involves the computation of many analogical equations, and a statistic as simple as the frequency of a solution often suffices to separate good from spurious solutions.

2.2 Analogical Solver

Lepage (1998) proposed an algorithm for computing the solutions of a formal analogical equation $[\textit{\textbf{A}} : \textit{\textbf{B}} = \textit{\textbf{C}} : ?]$. We implemented a variant of this algorithm which requires to compute two edit-distance tables, one between A and B and one between A and C. Since we are looking for subsequences of B and C not present in A, insertion cost is null. Once this is done, the algorithm synchronizes the alignments defined by the paths of minimum cost in each table. Intuitively, the synchronization of two alignments (one between A and B, and one between A and C) consists in composing in the correct order subsequences of the strings B and C that are not in A. We refer the reader to (Lepage, 1998) for the intricacies of this process which is illustrated in Figure 1 for the analogical equation [*even* : *usual* = *unevenly* : ?]. In this example, there are 681 different paths that align *even* and *usual* (with a cost of 4), and 1 path which aligns *even* with *unevenly* (with a cost of 0). This results in 681 synchronizations which generate 15 different solutions, among which only *unusually* is a legitimate word-form.

In practice, since the number of minimum-cost paths may be exponential in the size of the strings being aligned, we consider the synchronization of a maximum of M best paths in each edit-distance table. The worst-case complexity of our analogical solver is $O([|A| \times (|B| + |C|)] + [M^2 \times (|A| + ins(B, C))])$, where the first term corresponds to the computation of the two edit-distance tables,

[2] In Turney's work (Turney, 2006), a stem designates the first two words of a proportional analogy.

$\boxed{\boldsymbol{4}}$	4	4	4	4	4	n	4	4	3	3	2	1	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$
3	$\boxed{\boldsymbol{3}}$	3	3	3	3	e	3	3	3	2	1	$\boxed{0}$	0	0	0
2	$\boxed{\boldsymbol{2}}$	$\boxed{2}$	2	2	2	v	2	2	2	1	$\boxed{0}$	0	0	0	0
1	***1***	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	e	1	1	1	$\boxed{0}$	0	0	0	0	0
0	***0***	***0***	***0***	***0***	$\boxed{\boldsymbol{0}}$		$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	0	0	0	0	0	0
l	a	u	s	u	◁		▷	u	n	e	v	e	n	l	y

				e	v	e	n
u	**s**	**u**	**a**				**l**

		e	v	e	n		
u	**n**	e	v	e	n	**l**	**y**

⇒usua-un-l-ly

e				v		e	n
	u	**s**	**u**		**a**		**l**

		e	v	e	n		
u	**n**	e	v	e	n	**l**	**y**

⇒un-usu-a-l-ly

Figure 1: The top table reports the edit-distance tables computed between *even* and *usual* (left part), and *even* and *unevenly* (right part). The bottom part of the figure shows 2 of the 681 synchronizations computed while solving the equation [*even* : *usual* = *unevenly* : ?]. The first one corresponds to the path marked in bold italics and leads to a spurious solution; the second leads to a legitimate solution and corresponds to the path shown as squares.

and the second one corresponds to the maximum time needed to synchronize them. $|X|$ denotes the length, counted in characters of the string X, whilst $ins(B, C)$ stands for the number of characters of B and C not belonging to A. Given the typical length of the strings we consider in this study, our solver is quite efficient.[3]

Stroppa and Yvon (2005) described a generalization of this algorithm which can solve a formal analogical equation by composing two finite-state transducers.

2.3 Application to Lexical Enrichment

Analogical inference can be applied to the task of extending an existing bilingual lexicon (or transfer table) with new entries. In this study, we focus on a particular enrichment task: the one of translating valid words or phrases that were not encountered at training time.

A simple example of how our approach translates unknown words is illustrated in Figure 2 for the (unknown) French word *futilité*. In this example, translations is inferred by commuting plural and singular words. The inference process lazily captures the fact that English plural nouns ending in *-ies* usually correspond to singular nouns ending in *-y*.

Step 1	source (French) stems
[*activités* : *activité* = *futilités* : *futilité*]	
[*hostilités* : *hostilité* = *futilités* : *futilité*]	...
Step 2a	projection by lexicon look-up
activités↔*actions*	*hostilité*↔*hostility*
hostilités↔*hostilities*	*activité*↔*action*
futilités↔*trivialities,gimmicks*	...
Step 2b	target (English) resolution
[*actions* : *action* = *gimmicks* : ?]	⇒ gimmick
[*hostilities* : *hostility* = *trivialities* : ?]	⇒ triviality
[*hobbies* : *hobby* = *trivialities* : ?]	⇒ triviality
Step 3	selection of target candidates
⟨*triviality*, *2*⟩, ⟨*gimmick*, *1*⟩, ...	

Figure 2: Illustration of the analogical inference procedure applied to the translation of the unknown French word *futilité*.

Formally, we are given a training corpus $\mathcal{L} = \{\langle S_1, T_1\rangle, \ldots, \langle S_N, T_N\rangle\}$ which consists of a collection of N bilingual lexicon entries $\langle S_i, T_i\rangle$. The input space is in our case the space of possible source words, while the output space is the set of possible target words. We define:

$$\forall X \equiv \langle S, T\rangle,\ I(X) = S \text{ and } O(X) = T$$

Given an unknown source word-form S, Step 1 of the inference process consists in identifying source stems which have S as a solution:[4]

$$\mathcal{E}_{\mathcal{I}}(S) = \{\langle i, j, k\rangle \in [1, N]^3 \mid [S_i : S_j = S_k : S]\}.$$

During Step 2a, each source stem belonging to $\mathcal{E}_{\mathcal{I}}(S)$ is projected form by form into (potentially several) stems in the output space, thanks to an operator $proj$ that will be defined shortly:

$$\begin{aligned}&\mathcal{E}_{\langle i,j,k\rangle}(S) = \{T \mid [U : V = W : T]\} \text{ where}\\ &(U, V, W) \in (proj_{\mathcal{L}}(S_i) \times proj_{\mathcal{L}}(S_j) \times proj_{\mathcal{L}}(S_k)).\end{aligned}$$

[3] Several thousands of equations solved within one second.

[4] All strings in a stem must be different, otherwise, it can be shown that all source words would be considered.

During Step 2b, each solution to those output stems is collected in $\mathcal{E}_{\mathcal{O}}(S)$ along with its associated frequency:

$$\mathcal{E}_{\mathcal{O}}(S) = \bigcup_{\langle i,j,k\rangle \in \mathcal{E}_{\mathcal{I}}(S)} \mathcal{E}_{\langle i,j,k\rangle}(S).$$

Step 3 selects from $\mathcal{E}_{\mathcal{O}}(S)$ one or several solutions. We use frequency as criteria to sort the generated solutions. The projection mechanism we resort to in this study simply is a lexicon look-up:

$$proj_{\mathcal{L}}(S) = \{T \mid \langle S, T\rangle \in \mathcal{L}\}.$$

There are several situations where this inference procedure will introduce noise. First, both source and target analogical equations can lead to spurious solutions. For instance, $[$*show* : *showing* = *eating* : ?$]$ will erroneously produce *eatinging*. Second, an error in the original lexicon may introduce as well erroneous target word-forms. For instance, when translating the German word *proklamierung*, by making use of the analogy $[$*formalisiert* : *formalisierung* = *proklamiert* : *proklamierung*$]$, the English equation $[$*formalised* : *formalized* = *sets* : ?$]$ will be considered if it happens that *proklamiert*↔*sets* belongs to $\mathcal{L}$; in which case, *zets* will be erroneously produced.

We control noise in several ways. The source word-forms we generate are filtered by imposing that they belong to the input space. We also use a (large) target vocabulary to eliminate spurious target word-forms (see Section 3). More importantly, since we consider many analogical equations when translating a word-form, spurious analogical solutions tend to appear less frequently than ones arising from paradigmatic commutations.

2.4 Practical Considerations

Searching for $\mathcal{E}_{\mathcal{I}}(S)$ is an operation which requires solving a number of (source) analogical equations cubic in the size of the input space. In many settings of interest, including ours, this is simply not practical. We therefore resort to two strategies to reduce computation time. The first one consists in using the analogical equations in a generative mode. Instead of searching through the set of stems $\langle S_i, S_j, S_k\rangle$ that have for solution the unknown source word-form S, we search for all pairs (S_i, S_j) to the solutions of $[S_i : S_j = S :?]$ that are valid word-forms of the input space. Note that this is an exact method which follows from the property (Lepage, 1998):

$$[A : B = C : D] \equiv [B : A = D : C]$$

This leaves us with a quadratic computation time which is still intractable in our case. Therefore, we apply a second strategy which consists in computing the analogical equations $[S_i : S_j = S :?]$ for the only words S_i and S_j close enough to S. More precisely, we enforce that $S_i \in v_\delta(S)$ and that $S_j \in v_\beta(S_i)$ for a neighborhood function $v_\gamma(A)$ of the form:

$$v_\gamma(A) = \{B \mid f(B, A) \leq \gamma\}$$

where f is a distance; we used the edit-distance in this study (Levenshtein, 1966). Note that the second strategy we apply is only a heuristic.

3 Resources

In this work, we are concerned with one concrete problem a machine translation system must face: the one of translating unknown words. We are further focusing on the shared task of the workshop on Statistical Machine Translation, which took place last year (Koehn and Monz, 2006) and consisted in translating Spanish, German, and French texts from and to English. For some reasons, we restricted ourselves to translating only into English. The training material available is coming from the *Europarl* corpus. The test material was divided into two parts.[5] The first one (hereafter called `test-in`) is composed of 2 000 sentences from European parliament debates. The second part (called `test-out`) gathers 1 064 sentences[6] collected from editorials of the Project Syndicate website.[7] The main statistics pertinent to our study are summarized in Table 1.

A rough analysis of the 441 different unknown words encountered in the French test sets reveals that 54 (12%) of them contain at least one digit (years, page numbers, law numbers, etc.), 83 (20%) are proper names, 37 (8%) are compound words, 18 (4%) are foreign words (often Latin or Greek

[5] The participants were not aware of this.

[6] We removed 30 sentences which had encoding problems.

[7] `http://www.project-syndicate.com`

	French		Spanish		German	
test-	in	out	in	out	in	out
|unknown|	180	265	233	292	469	599
oov%	0.26	1.22	0.38	1.37	0.84	2.87

Table 1: Number of different (source) test words not seen at training time, and out-of-vocabulary rate expressed as a percentage (oov%).

words), 7 words are acronyms, and 4 are tokenization problems. The 238 other words (54%) are ordinary words.

We considered different lexicons for testing our approach. These lexicons were derived from the training material of the shared task by training with GIZA++ (Och and Ney, 2000) —default settings— two transfer tables (source-to-target and the reverse) that we intersected to remove some noise.

In order to investigate how sensitive our approach is to the amount of training material available, we varied the size of our lexicon $\mathcal{L}_T$ by considering different portions of the training corpus ($T = 5\,000$, 10 000, 100 000, 200 000, and 500 000 pairs of sentences). The lexicon trained on the full training material (688 000 pairs of sentences), called $\mathcal{L}_{ref}$ hereafter, is used for validation purposes. We kept (at most) the 20 best associations of each source word in these lexicons. In practice, because we intersect two models, the average number of translations kept for each source word is lower (see Table 2).

Last, we collected from various target texts (English here) we had at our disposal, a vocabulary set $\mathcal{V}$ gathering 466 439 words, that we used to filter out spurious word-forms generated by our approach.

4 Experiments

4.1 Translating Unknown Words

For the three translation directions (from Spanish, German, and French into English), we applied the analogical reasoning to translate the (non-numerical) source words of the test material, absent from $\mathcal{L}_T$. Examples of translations produced by analogical inference are reported in Figure 3, sorted by decreasing order of times they have been generated.

anti-agricole ◇ *(anti-farm,5) (anti-agricultural,3) (anti-rural,3) (anti-farming,3) (anti-farmer,3)*

fleurie ◇ *(flourishing,5) (flourished,4) (flourish,1)*

futilité ◇ *(trivialities,27) (triviality,14) (futile,9) (meaningless,9) (futility,4) (meaninglessness,4) (superfluous,2) (unwieldy,2) (unnecessary,2) (uselessness,2) (trivially,1)* ***(tie,1)*** *(trivial,1)*

butoir ◇ *(deadline,42) (deadlines,33)* **(blows,1)**

court-circuitant ◇ *(bypassing,13) (bypass,12) (bypassed,5) (bypasses,1)*

xviie ◇ *(xvii,18)* ***(sixteenth,3) (eighteenth,1)***

Figure 3: Candidate translations inferred from $\mathcal{L}_{200\,000}$ and their frequency. The candidates reported are those that have been intersected with $\mathcal{V}$. Translations in bold are clearly erroneous.

4.1.1 Baselines

We devised two baselines against which we compared our approach (hereafter ANALOG). The first one, BASE1, simply proposes as translations the target words in the lexicon $\mathcal{L}_T$ which are the most similar (in the sense of the edit-distance) to the unknown source word. Naturally, this approach is only appropriate for pairs of languages that share many cognates (*i.e.*, *docteur* → *doctor*). The second baseline, BASE2, is more sensible and more closely corresponds to our approach. We first collect a set of source words that are close-enough (according to the edit-distance) to the unknown word. Those source words are then projected into the output space by simple bilingual lexicon look-up. So for instance, the French word *demanda* will be translated into the English word *request* if the French word *demande* is in $\mathcal{L}_T$ and that *request* is one of its sanctioned translations.

Each of these baselines is tested in two variants. The first one ($_{id}$), which allows a direct comparison, proposes as many translations as ANALOG does. The second one ($_{10}$) proposes the first 10 translations of each unknown word.

4.1.2 Automatic Evaluation

Evaluating the quality of translations requires to inspect lists of words each time we want to test a variant of our approach. This cumbersome process not only requires to understand the source language,

$\mathcal{L}_T$	5 000		10 000		50 000		100 000		200 000		500 000	
	p%	r%	p%	r%	p%	r%	p%	r%	p%	r%	p%	r%
	`test-in`											
ANALOG	51.4	30.7	55.3	44.4	58.8	64.3	58.2	65.1	59.4	65.2	30.4	67.6
BASE1$_{id}$	31.6	30.7	32.3	44.4	24.7	64.3	20.3	65.1	20.9	65.2	8.7	67.6
BASE2$_{id}$	34.5	30.7	37.1	44.4	39.0	64.3	37.8	65.1	34.4	65.2	56.5	67.6
BASE1$_{10}$	26.7	100.0	28.3	100.0	23.9	100.0	20.0	100.0	16.6	100.0	11.8	100.0
BASE2$_{10}$	26.3	100.0	30.8	100.0	29.3	100.0	27.6	100.0	24.9	100.0	55.9	100.0
unk	$[3\,171\,,\,9.1]$		$[2\,245\,,\,7.7]$		$[754\,,\,4.0]$		$[456\,,\,2.9]$		$[253\,,\,2.0]$		$[34\,,\,1.2]$	
	`test-out`											
ANALOG	52.8	28.9	55.3	42.5	52.9	68.8	54.7	74.6	55.7	81.0	43.3	88.2
BASE1$_{id}$	28.0	28.9	29.0	42.5	27.3	68.8	23.1	74.6	26.8	81.0	22.7	88.2
BASE2$_{id}$	32.9	28.9	35.0	42.5	32.5	68.8	35.9	74.6	40.8	81.0	59.1	88.2
BASE1$_{10}$	24.7	100.0	25.9	100.0	25.1	100.0	20.9	100.0	25.2	100.0	25.0	100.0
BASE2$_{10}$	21.7	100.0	26.4	100.0	27.2	100.0	29.4	100.0	33.6	100.0	57.9	100.0
unk	$[2\,270\,,\,8.2]$		$[1\,701\,,\,6.9]$		$[621\,,\,3.4]$		$[402\,,\,2.4]$		$[226\,,\,1.8]$		$[76\,,\,1.4]$	

Table 2: Performance of the different approaches on the French-to-English direction as a function of the number T of pairs of sentences used for training $\mathcal{L}_T$. A pair $[n\,,\,t]$ in lines labeled by *unk* stands for the number of words to translate, and the average number of their translations in $\mathcal{L}_{ref}$.

but happens to be in practice a delicate task. We therefore decided to resort to an automatic evaluation procedure which relies on $\mathcal{L}_{ref}$, a bilingual lexicon which entries are considered correct.

We translated all the words of $\mathcal{L}_{ref}$ absent from $\mathcal{L}_T$. We evaluated the different approaches by computing *response* and *precision* rates. The response rate is measured as the percentage of words for which we do have at least one translation produced (correct or not). The precision is computed in our case as the percentage of words for which at least one translation is sanctioned by $\mathcal{L}_{ref}$. Note that this way of measuring response and precision is clearly biased toward translation systems that can hypothesize several candidate translations for each word, as statistical systems usually do. The reason of this choice was however guided by a lack of precision of the reference we anticipated, a point we discuss in Section 4.1.3.

The figures for the French-to-English direction are reported in Table 2. We observe that the ratio of unknown words that get a translation by ANALOG is clearly impacted by the size of the lexicon $\mathcal{L}_T$ we use for computing analogies: the larger the better. This was expected since the larger a lexicon is, the higher the number of source analogies that can be made and consequently, the higher the number of analogies that can be projected onto the output space. The precision of ANALOG is rather stable across variants and ranges between 50% to 60%.

The second observation we make is that the baselines perform worse than ANALOG in all but the $\mathcal{L}_{500\,000}$ cases. Since our baselines propose translations to each source word, their response rate is maximum. Their precision, however, is an issue. Expectedly, BASE1 is the worst of the two baselines. If we arbitrarily fix the response rate of BASE2 to the one of ANALOG, the former approach shows a far lower precision (*e.g.,* 34.4 against 59.4 for $\mathcal{L}_{200\,000}$). This not only indicates that analogical learning is handling unknown words better than BASE2, but as well, that a combination of both approaches could potentially yield further improvements.

A last observation concerns the fact that ANALOG performs equally well on the out-domain material. This is very important from a practical point of view and contrasts with some related work we discuss in Section 5.

At first glance, the fact that BASE2 outperforms ANALOG on the larger training size is disappointing. After investigations, we came to the conclusion that this is mainly due to two facts. First, the number of unknown words on which both systems were tested is rather low in this particular case (*e.g.,* 34 for the in-domain corpus). Second, we noticed a deficiency of the reference lexicon $\mathcal{L}_{ref}$ for many of those words. After all, this is not surprising since the words unseen in the 500 000 pairs of training sentences, but encountered in the full training corpus (688 000 pairs) are likely to be observed only a few times, therefore weakening the associations automatically acquired for these entries. We evaluate that a third of the reference translations were wrong in this setting, which clearly raises some doubts on our automatic evaluation procedure in this case.

The performance of ANALOG across the three language pairs are reported in Table 3. We observe a drop of performance of roughly 10% (both in precision and response) for the German-to-English translation direction. This is likely due to the heuristic procedure we apply during the search for stems, which is not especially well suited for handling compound words that are frequent in German.

We observe that for Spanish- and German-to-English translation directions, the precision rate tends to decrease for larger values of T. One explanation for that is that we consider all analogies equally likely in this work, while we clearly noted that some are spurious ones. With larger training material, spurious analogies become more likely.

	French		Spanish		German	
T	p%	r%	p%	r%	p%	r%
5	51.4	30.7	52.8	30.3	49.3	23.1
10	55.3	44.4	52.0	45.2	47.6	33.3
50	58.8	64.3	54.0	66.5	44.6	53.2
100	58.2	65.1	53.9	69.1	45.8	55.6
200	59.4	65.2	46.4	71.8	43.0	59.2

Table 3: Performance across language pairs measured on `test-in`. The number T of pairs of sentences used for training $\mathcal{L}_T$ is reported in thousands.

We measured the impact the translations produced by ANALOG have on a state-of-the-art phrase-based translation engine, which is described in (Patry et al., 2006). For that purpose, we extended a phrase-table with the first translation proposed by ANALOG or BASE2 for each unknown word of the test material. Results in terms of word-error-rate (WER) and BLEU score (Papineni et al., 2002) are reported in Table 4 for those sentences that contain at least one unknown word. Small but consistent improvements are observed for both metrics with ANALOG. This was expected, since the original system simply leaves the unknown words untranslated. What is more surprising is that the BASE2 version slightly underperforms the baseline. The reason is that some unknown words that should appear unmodified in a translation, often get an erroneous translation by BASE2. Forcing BASE2 to propose a translation for the same words for which ANALOG found one, slightly improves the figures (BASE2_{id}).

	French		Spanish		German	
	WER	BLEU	WER	BLEU	WER	BLEU
base	61.8	22.74	54.0	27.00	69.9	18.15
+BASE2	61.8	22.72	54.2	26.89	70.3	18.05
+BASE2_{id}	61.7	22.81	54.1	27.01	70.1	18.14
+ANALOG	61.6	22.90	53.7	27.27	69.7	18.30
sentences	387		452		814	

Table 4: Translation quality produced by our phrase-based SMT engine (base) with and without the first translation produced by ANALOG, BASE2, or BASE2_{id} for each unknown word.

4.1.3 Manual Evaluation

As we already mentioned, the lexicon used as a reference in our automatic evaluation procedure is not perfect, especially for low frequency words. We further noted that several words receive valid translations that are not sanctioned by $\mathcal{L}_{ref}$. This is for instance the case of the examples in Figure 4, where *circumventing* and *fellow* are arguably legitimate translations of the French words *contournant* and *concitoyen*, respectively. Note that in the second example, the reference translation is in the plural form while the French word is not.

Therefore, we conducted a manual evaluation of the translations produced from $\mathcal{L}_{100\,000}$ by ANALOG and BASE2 on the 127 French words of the corpus `test-in`[8] unknown of $\mathcal{L}_{ref}$. Those are the non-numerical unknown words the participating systems in the shared task had to face in the

[8] We did not notice important differences between `test-in` and `test-out`.

contournant	(*49 candidates*)
ANALOG ◇ (circumventing,55) (undermining,20) (evading,19) (circumvented,17) (overturning,16) (circumvent,15) (circumvention,15) (bypass,13) (evade,13) (skirt,12)	
$\mathcal{L}_{ref}$ ◇ **skirting**, **bypassing**, by-pass, overcoming	

concitoyen	(*24 candidates*)
ANALOG ◇ (citizens,26) (fellow,26) (**fellow-citizens**,26) (people,26) (citizen,23) (fellow-citizen,21) (fellows,5) (peoples,3) (civils,3) (fellowship,2)	
$\mathcal{L}_{ref}$ ◇ **fellow-citizens**	

Figure 4: 10 best ranked candidate translations produced by ANALOG from $\mathcal{L}_{200\,000}$ for two unknown words and their sanctioned translations in $\mathcal{L}_{ref}$. Words in bold are present in both the candidate and the reference lists.

in-domain part of the test material. 75 (60%) of those words received at least one valid translation by ANALOG while only 63 (50%) did by BASE2. Among those words that received (at least) one valid translation, 61 (81%) were ranked first by ANALOG against only 22 (35%) by BASE2. We further observed that among the 52 words that did not receive a valid translation by ANALOG, 38 (73%) did not receive a translation at all. Those untranslated words are mainly proper names (*bush*), foreign words (*munere*), and compound words (*rhénanie-du-nord-westphalie*), for which our approach is not especially well suited.

We conclude from this informal evaluation that 80% of ordinary unknown words received a valid translation in our French-to-English experiment, and that roughly the same percentage had a valid translation proposed in the first place by ANALOG.

4.2 Translating Unknown Phrases

Our approach is not limited to translate solely unknown words, but might serve as well to enrich existing entries in a lexicon. For instance, low-frequency words, often poorly handled by current statistical methods, could receive useful translations. This is illustrated in Figure 5 where we report the best candidates produced by ANALOG for the French word *invitées*, which appears 7 times in the 200 000 first pairs of the training corpus. Interestingly, ANALOG produced the candidate *guest* which corresponds to a legitimate meaning of the French word that was absent in the training data.

invitée	(*61 candidates*)
ANALOG ◇ (invited,135) (requested,92) (called,77) (**urged**,75) (guest,72) (**asked**,47) (request,43) (invites,27) (invite,26) (urge,26)	
$\mathcal{L}_{200\,000}$ ◇ **asked**, generate, **urged**	

Figure 5: 10 best candidates produced by ANALOG for the low-frequency French word *invitées* and its translations in $\mathcal{L}_{200\,000}$.

Because it can treat separators as any other character, ANALOG is not bounded to translate only words. As a proof of concept, we applied analogical reasoning to translate those source sequences of at most 5 words in the test material that contain an unknown word. Since there are many more sequences than there are words, the input space in this experiment is far larger, and we had to resort to a much more aggressive pruning technique to find the stems of the sequences to be translated.

expulsent ◇ (expelling,36) (expel,31) (are expelling,23) (are expel,10)
focaliserai ◇ (focus,10) (focus solely,9) (concentrate all,9) (will focus,9) (will placing,9)
dépasseront ◇ (will exceed,4) (exceed,3) (will be exceed,3) (we go beyond,2) (will be exceeding,2)
non-réussite *de* ◇ (lack of success for,4) (lack of success of,4) (lack of success,4)
que vous ***subissez*** ◇ (you are experiencing,2)

Figure 6: Examples of translations produced by ANALOG where the input (resp. output) space is defined by the set of source (resp. target) word sequences. Words in bold are unknown.

We applied the automatic evaluation procedure described in Section 4.1.2 for the French-to-English translation direction, with a reference lexicon being this time the phrase table acquired on the full training material.[9] The response rate in this experiment is particularly low since only a tenth of the sequences

[9] This model contains 1.5 millions pairs of phrases.

received (at least) a translation by ANALOG. Those are short sequences that contain at most three words, which clearly indicates the limitation of our pruning strategy. Among those sequences that received at least one translation, the precision rate is 55%, which is consistent with the rate we measured while translating words.

Examples of translations are reported in Figure 6. We observe that single words are not contrived anymore to be translated by a single word. This allows to capture *1:n* relations such as *dépasseront*↔*will exceed*, where the future tense of the French word is adequately rendered by the modal *will* in English.

5 Related Work

We are not the first to consider the translation of unknown words or phrases. Several authors have for instance proposed approaches for translating proper names and named entities (Chen et al., 1998; Al-Onaizan and Knight, 2002). Our approach is complementary to those ones.

Recently and more closely related to the approach we described, Callison-Burch et al. (2006) proposed to replace an unknown phrase in a source sentence by a paraphrase. Paraphrases in their work are acquired thanks to a word alignment computed over a large external set of bitexts. One important difference between their work and ours is that our approach does not require additional material.[10] Indeed, they used a rather idealistic set of large, homogeneous bitexts (European parliament debates) to acquire paraphrases from. Therefore we feel our approach is more suited for translating "low density" languages and languages with a rich morphology.

Several authors considered as well the translation of new words by relying on distributional collocational properties computed from a huge non-parallel corpus (Rapp, 1999; Fung and Yee, 1998; Takaaki and Matsuo, 1999; Koehn and Knight, 2002). Even if admittedly non-parallel corpora are easier to acquire than bitexts, this line of work is still heavily dependent on huge external resources.

Most of the analogies made at the word level in our study are capturing morphological information. The use of morphological analysis in (statistical) machine translation has been the focus of several studies, (Nießen, 2002) among the first. Depending on the pairs of languages considered, gains have been reported when the training material is of modest size (Lee, 2004; Popovic and Ney, 2004; Goldwater and McClosky, 2005). Our approach does not require any morphological knowledge of the source, the target, or both languages. Admittedly, several unsupervised morphological induction methodologies have been proposed, *e.g.*, the recent approach in Freitag (2005). In any case, as we have shown, ANALOG is not bounded to treat only words, which we believe to be at our advantage.

6 Discussion and Future Work

In this paper, we have investigated the appropriateness of analogical learning to handle unknown words in machine translation. On the contrary to several lines of work, our approach does not rely on massive additional resources but capitalizes instead on an information which is inherently pertaining to the language. We measured that roughly 80% of ordinary unknown French words can receive a valid translation into English with our approach.

This work is currently being developed in several directions. First, we are investigating why our approach remains silent for some words or phrases. This will allow us to better characterize the limitations of ANALOG and will hopefully lead us to design a better strategy for identifying the stems of a given word or phrase. Second, we are investigating how a systematic enrichment of a phrase-transfer table will impact a phrase-based statistical machine translation engine. Last, we want to investigate the training of a model that can learn regularities from the analogies we are making. This would relieve us from requiring the training material while translating, and would allow us to compare our approach with other methods proposed for unsupervised morphology acquisition.

Acknowledgement We are grateful to the anonymous reviewers for their useful suggestions and to Pierre Poulin for his fruitful comments. This study has been partially funded by NSERC.

[10] We do use a target vocabulary list to filter out spurious analogies, but we believe we could do without. The frequency with which we generate a string could serve to decide upon its legitimacy.

References

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proc. of the 40th ACL*, pages 400–408, Philadelphia, Pennsylvania, USA.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proc. of HLT-NAACL*, pages 17–24, New York City, USA.

Hsin-Hsi Chen, Sheng-Jie Hueng, Yung-Wei Ding, and Shih-Chung Tsai. 1998. Proper name translation in cross-language information retrieval. In *Proc. of the 17th COLING*, pages 232–236, Montreal, Québec, Canada.

Matthias Eck and Chiori Hori. 2005. Overview of the IWSLT 2005 evaluation campaign. In *International Workshop on Spoken Language Translation (IWSLT)*, Pittsburgh, Pennsylvania, USA.

Dayne Freitag. 2005. Morphology induction from term clusters. In *Proc. of the 9th CoNLL*, pages 128–135, Ann Arbor, Michigan, USA.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proc. of the 36th ACL*, pages 414–420, San Francisco, California, USA.

Dedre Gentner, Keith J. Holyoak, and Boicho N. Konikov. 2001. *The Analogical Mind*. The MIT Press, Cambridge, Massachusetts, USA.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proc. of HLT-EMNLP*, pages 676–683, Vancouver, British Columbia, Canada.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proc. of the ACL Workshop on Unsupervised Lexical Acquisition*, pages 9–16, Philadelphia, Pennsylvania, USA.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proc. of the 10th EACL*, pages 187–193, Budapest, Hungary.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between european languages. In *Proc. of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 102–121, New York City, USA.

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proc. of HLT-NAACL*, Boston, Massachusetts, USA.

Yves Lepage and Etienne Denoual. 2005. ALEPH: an EBMT system based on the preservation of proportionnal analogies between sentences across languages. In *Proc. of IWSLT*, Pittsburgh, Pennsylvania, USA.

Yves Lepage. 1998. Solving analogies on words: an algorithm. In *Proc. of COLING-ACL*, pages 728–734, Montreal, Québec, Canada.

Yves Lepage. 2003. *De l'analogie rendant compte de la commutation en linguistique*. Ph.D. thesis, Université Joseph Fourier, Grenoble, France.

Vladimir. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.*, 6:707–710.

Fabienne Moreau, Vincent Claveau, and Pascale Sébillot. 2007. Automatic morphological query expansion using analogy-based machine learning. In *Proc. of the 29th ECIR*, Roma, Italy.

Sonja Nießen. 2002. *Improving Statistical Machine Translation using Morpho-syntactic Information*. Ph.D. thesis, RWTH, Aachen, Germany.

Franz-Joseph Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proc. of the 38th ACL*, pages 440–447, Hong Kong, China.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of the 40th ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Alexandre Patry, Fabrizo Gotti, and Philippe Langlais. 2006. Mood at work: Ramses versus Pharaoh. In *Proc. of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 126–129, New York City, USA.

Maja Popovic and Hermann Ney. 2004. Towards the use of word stems and suffixes for statistical machine translation. In *Proc. of the 4th LREC*, pages 1585–1588, Lisbon, Portugal.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proc. of the 37th ACL*, pages 519–526, College Park, Maryland, USA.

Nicolas Stroppa and François Yvon. 2005. An analogical learner for morphological analysis. In *Proc. of the 9th CoNLL*, pages 120–127, Ann Arbor, Michigan, USA.

Tanaka Takaaki and Yoshihiro Matsuo. 1999. Extraction of translation equivalents from non-parallel corpora. In *Proc. of the 8th TMI*, pages 109–119, Chester, England.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, Sept.

A Probabilistic Approach to Diachronic Phonology

Alexandre Bouchard-Côté* **Percy Liang*** **Thomas L. Griffiths**† **Dan Klein***
*Computer Science Division †Department of Psychology
University of California at Berkeley
Berkeley, CA 94720

Abstract

We present a probabilistic model of diachronic phonology in which individual word forms undergo stochastic edits along the branches of a phylogenetic tree. Our approach allows us to achieve three goals with a single unified model: (1) reconstruction of both ancient and modern word forms, (2) discovery of general phonological changes, and (3) selection among different phylogenies. We learn our model using a Monte Carlo EM algorithm and present quantitative results validating the model.

1 Introduction

Modeling how languages change phonologically over time (diachronic phonology) is a central topic in historical linguistics (Campbell, 1998). The questions involved range from reconstruction of ancient word forms, to the elucidation of phonological drift processes, to the determination of phylogenetic relationships between languages. However, this problem has received relatively little attention from the computational community. What work there is has focused on the reconstruction of phylogenies on the basis of a Boolean matrix indicating the properties of words in different languages (Gray and Atkinson, 2003; Evans et al., 2004; Ringe et al., 2002; Nakhleh et al., 2005).

In this paper, we present a novel framework, along with a concrete model and experiments, for the probabilistic modeling of diachronic phonology. We focus on the case where the words are etymological cognates across languages, e.g. French *faire* and Spanish *hacer* from Latin *facere* (to do). Given this information as input, we learn a model acting at the level of individual phoneme sequences, which can be used for reconstruction and prediction, Our model is fully generative, and can be used to reason about a variety of types of information. For example, we can observe a word in one or more modern languages, say French and Spanish, and query the corresponding word form in another language, say Italian. This kind of lexicon-filling has applications in machine translation. Alternatively, we can also reconstruct ancestral word forms or inspect the rules learned along each branch of a phylogeny to identify salient patterns. Finally, the model can be used as a building block in a system for inferring the topology of phylogenetic trees. We discuss all of these cases further in Section 4.

The contributions of this paper are threefold. First, the approach to modeling language change at the phoneme sequence level is new, as is the specific model we present. Second, we compiled a new corpus[1] and developed a methodology for quantitatively evaluating such approaches. Finally, we describe an efficient inference algorithm for our model and empirically study its performance.

1.1 Previous work

While our word-level model of phonological change is new, there have been several computational investigations into diachronic linguistics which are relevant to the present work.

The task of reconstructing phylogenetic trees

[1] `nlp.cs.berkeley.edu/pages/historical.html`

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 887–896, Prague, June 2007. ©2007 Association for Computational Linguistics

for languages has been studied by several authors. These approaches descend from *glottochronology* (Swadesh, 1955), which views a language as a collection of shared cognates but ignores the structure of those cognates. This information is obtained from manually curated cognate lists such as the data of Dyen et al. (1997).

As an example of a cognate set encoding, consider the meaning "eat". There would be one column for the cognate set which appears in French as *manger* and Italian as *mangiare* since both descend from the Latin *mandere* (to chew). There would be another column for the cognate set which appears in both Spanish and Portuguese as *comer*, descending from the Latin *comedere* (to consume). If this were the only data, algorithms based on this data would tend to conclude that French and Italian were closely related and that Spanish and Portuguese were equally related. However, the cognate set representation has several disadvantages: it does not capture the fact that the cognate is closer between Spanish and Portuguese than between French and Spanish, nor do the resulting models let us conclude anything about the regular processes which caused these languages to diverge. Also, the existing cognate data has been curated at a relatively high cost. In our work, we track each word using an automatically obtained cognate list. While our cognates may be noisier, we compensate by modeling phonological changes rather than boolean mutations in cognate sets.

There has been other computational work in this broad domain. Venkataraman et al. (1997) describe an information theoretic measure of the distance between two dialects of Chinese. Like our approach, they use a probabilistic edit model as a formalization of the phonological process. However, they do not consider the question of reconstruction or inference in multi-node phylogenies, nor do they present a learning algorithm for such models.

Finally, for the specific application of cognate prediction in machine translation, essentially transliteration, there have been several approaches, including Kondrak (2002). However, the phenomena of interest, and therefore the models, are extremely different. Kondrak (2002) presents a model for learning "sound laws," general phonological changes governing two completely observed aligned cognate lists. His model can be viewed as a special case of ours using a simple two-node topology.

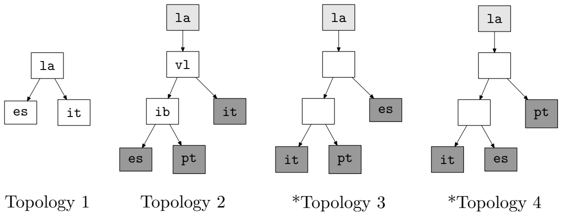


Figure 1: Tree topologies used in our experiments. *Topology 3 and *Topology 4 are incorrect evolutionary tree used for our experiments on the selection of phylogenies (Section 4.4).

There is also a rich literature (Huelsenbeck et al., 2001) on the related problems of evolutionary biology. A good reference on the subject is Felsenstein (2003). In particular, Yang and Rannala (1997), Mau and Newton (1997) and Li et al. (2000) each independently presented a Bayesian model for computing posteriors over evolutionary trees. A key difference with our model is that independence across evolutionary sites is assumed in their work, while the evolution of the phonemes in our model depends on the environment in which the change occurs.

2 A model of phonological change

Assume we have a fixed set of *word types* (cognate sets) in our vocabulary V and a set of languages L. Each word type i has a *word form* w_{il} in each language $l \in L$, which is represented as a sequence of phonemes and might or might not be observed. The languages are arranged according to some tree topology T (see Figure 1 for examples). One might consider models that simultaneously induce the topology and cognate set assignments, but let us fix both for now. We discuss one way to relax this assumption and present experimental results in Section 4.4.

Our generative model (Figure 3) specifies a distribution over the word forms $\{w_{il}\}$ for each word type $i \in V$ and each language $l \in L$. The generative process starts at the root language and generates all the word forms in each language in a top-down manner. One appealing aspect about our model is that, at a high-level, it reflects the actual phonological process that languages undergo. However, important phenomena like lexical drift, borrowing, and other non-phonological changes are not modeled.

Our generative model can be summarized as follows:

```
For each word $i \in V$:
  $w_{i\text{ROOT}} \sim$ LanguageModel
For each branch $(k \to l) \in T$:
  $\theta_{k \to l} \sim$ Dirichlet($\alpha$)      [choose edit params.]
  For each word $i \in V$:
    $w_{il} \sim$ Edit($w_{ik}, \theta_{k \to l}$)      [sample word form]
```

In the remainder of this section, we describe each of the steps in the model.

2.1 Language model

For the distribution $w \sim$ LanguageModel, we used a simple bigram phoneme model. The phonemes were partitioned into *natural classes* (see Section 4 for details). A root word form consisting of n phonemes $x_1 \cdots x_n$ is generated with probability

$$p_{\text{lm}}(x_1) \prod_{j=2}^{n} p_{\text{lm}}(x_j \mid \text{NaturalClass}(x_{j-1})),$$

where p_{lm} is the distribution of the language model.

2.2 Edit model

The stochastic edit model $y \sim \text{Edit}(x, \theta)$ describes how a single old word form $x = x_1 \cdots x_n$ changes along one branch of the phylogeny with parameters θ to produce a new word form y. This process is parameterized by rule probabilities $\theta_{k \to l}$, which are specific to branch $(k \to l)$.

The generative process is as follows: for each phoneme x_i in the old word form, walking from left to right, choose a rule to apply. There are three types of rules: (1) *deletion* of the phoneme, (2) *substitution* with another phoneme (possibly the same one), or (3) *insertion* of another phoneme, either before or after the existing one. The probability of applying a rule depends on a *context* $(\text{NaturalClass}(x_{i-1}), \text{NaturalClass}(x_{i+1}))$. Figure 2 illustrates the edits on an example. The context-dependence allows us to represent phenomena such as the fact that s is likely to be deleted only in word-final contexts.

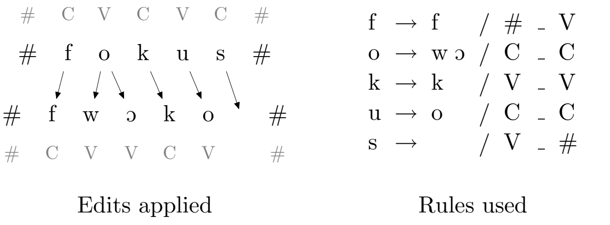


Figure 2: An example of edits that were used to transform the Latin word *FOCUS* (/fokus/) into the Italian word *fuoco* (/fwɔko/) (fire) along with the context-specific rules that were applied.

The edit model we have presented approximately encodes a limited form of classic rewrite-driven segmental phonology (Chomsky and Halle, 1968). One could imagine basing our model on more modern phonological theory, but the computational properties of the edit model are compelling, and it is adequate for many kinds of phonological change.

In addition to simple edits, we can model some classical changes that appear to be too complex to be captured by a single left-to-right edit model of this kind. For instance, bleeding and feeding arrangements occur when one phonological change introduces a new context, which triggers another phonological change, but the two cannot occur simultaneously. For example, vowel raising e → i / _ c might be needed before palatalization t → c / _ i. Instead of capturing such an interaction directly, we can break up a branch into two segments joined at an intermediate language node, conflating the concept of historically intermediate languages with the concept of intermediate stages in the application of sequential rules.

However, many complex processes are not well-represented by our basic model. One problematic case is chained shifts such as Grimm's law in Proto-Germanic or the Great Vowel Shift in English. To model such dependent rules, we would need to use a more complex prior distributions over the edit parameters. Another difficult case is prosodic changes, such as unstressed vowel neutralizations, which would require a representation of suprasegmental features. While our basic model does not account for these phenomena, extensions within the generative framework could capture such richness.

3 Learning and inference

We use a Monte Carlo EM algorithm to fit the parameters of our model. The algorithm iterates between a stochastic E-step, which computes recon-

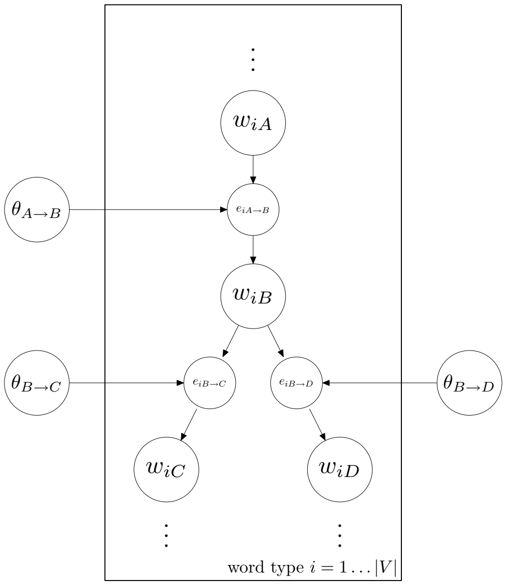


Figure 3: The graphical model representation of our model: θ are the parameters specifying the stochastic edits e, which govern how the words w evolve. The plate notation indicates the replication of the nodes corresponding to the evolving words.

structions based on the current edit parameters, and an M-step, which updates the edit parameters based on the reconstructions.

3.1 Monte Carlo E-step: sampling the edits

The E-step needs to produce expected counts of how many times each edit (such as o → ɔ) was used in each context. An exact E-step would require summing over all possible edits involving all languages in the phylogeny (all unobserved $\{e\}, \{w\}$ variables in Figure 3). Unfortunately, unlike in the case of HMMs and PCFGs, our model permits no tractable dynamic program to compute these counts exactly.

Therefore, we resort to a Monte Carlo E-step, where many samples of the edit variables are collected, and counts are computed based on these samples. Samples are drawn using Gibbs sampling (Geman and Geman, 1984): for each word form of a particular language w_{il}, we fix all other variables in the model and sample w_{il} along with its corresponding edits.

In the E-step, we fix the parameters, which renders the word types conditionally independent, just as in an HMM. Therefore, we can process each word type in turn without approximation.

First consider the simple 4-language topology in Figure 3. Suppose that the words in languages A, C and D are fixed, and we wish to infer the word at language B along with the three corresponding sets of edits (remember the edits fully determine the words). There are an exponential number of possible words/edits, but it turns out that we can exploit the Markov structure in the edit model to consider all such words/edits using dynamic programming, in a way broadly similar to the forward-backward algorithm for HMMs.

Figure 4 shows the lattice for the dynamic program. Each path connecting the two shaded endpoint states represents a particular word form for language B and a corresponding set of edits. Each node in the lattice is a state of the dynamic program, which is a 5-tuple $(i_A, i_C, i_D, c_1, c_2)$, where i_A, i_C and i_D are the cursor positions (represented by dots in Figure 4) in each of the word forms of A, C and D, respectively; c_1 is the natural class of the phoneme in the word form for B that was last generated; and c_2 corresponds to the phoneme that will be generated next.

Each state transition involves applying a rule to A's current phoneme (which produces 0–2 phonemes in B) and applying rules to B's new 0–2 phonemes. There are three types of rules (deletion, substitution, insertion), resulting in $3^0+3^2+3^4 = 91$ types of state transitions. For illustration, Figure 4 shows the simpler case where B only has one child C. Given these rules, the new state is computed by advancing the appropriate cursors and updating the natural classes c_1 and c_2. The weight of each transition $w(s \to t)$ is a product of the language model probability and the rule probabilities that were chosen.

For each state s, the dynamic program computes $W(s)$, the sum of the weights of all paths leaving s,

$$W(s) = \sum_{s \to t} w(s \to t) W(t).$$

To sample a path, we start at the leftmost state, choose the transition with probability proportional to its contribution in the sum for computing $W(s)$, and repeat until we reach the rightmost state.

We applied a few approximations to speed up the sampling of words, which reduced the running time by several orders of magnitude. For example, we pruned rules with low probability and restricted the

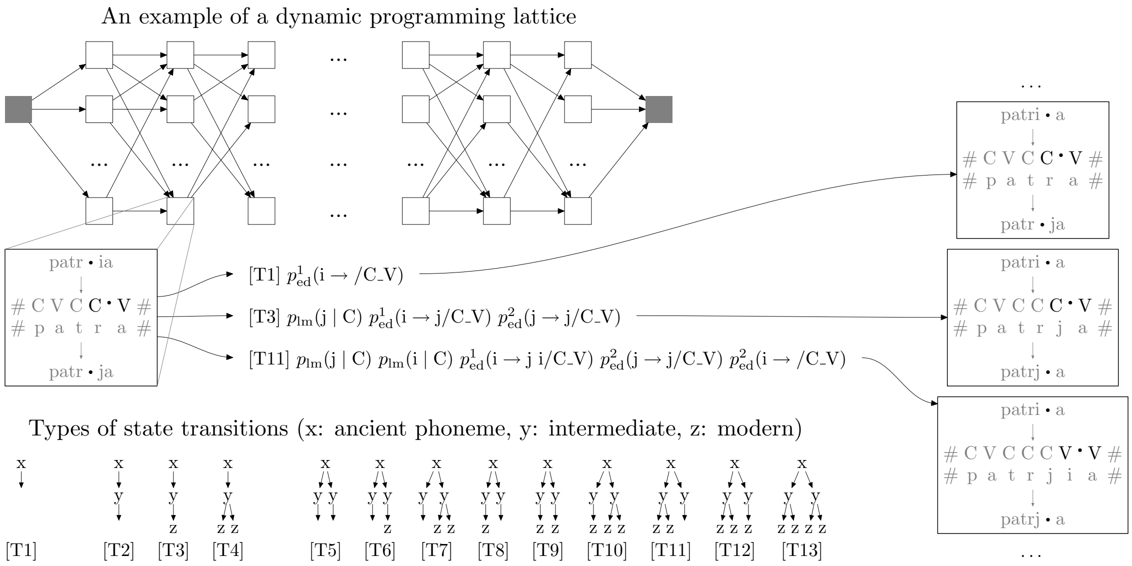

An example of a dynamic programming lattice
patr • ia
# C V C C • V #
# p a t r a #
patr • ja
[T1] p^1_{ed}(i → /C_V)
[T3] p_{lm}(j | C) p^1_{ed}(i → j/C_V) p^2_{ed}(j → j/C_V)
[T11] p_{lm}(j | C) p_{lm}(i | C) p^1_{ed}(i → j i/C_V) p^2_{ed}(j → j/C_V) p^2_{ed}(i → /C_V)
patri • a
# C V C C • V #
# p a t r a #
patr • ja
patri • a
# C V C C C • V #
# p a t r j a #
patrj • a
patri • a
# C V C C C V • V #
# p a t r j i a #
patrj • a
Types of state transitions (x: ancient phoneme, y: intermediate, z: modern)
[T1]
[T2]
[T3]
[T4]
[T5]
[T6]
[T7]
[T8]
[T9]
[T10]
[T11]
[T12]
[T13]

Figure 4: The dynamic program involved in sampling an intermediate word form given one ancient and one modern word form. One lattice node is expanded to show the dynamic program state (represented by the part not grayed out) and three of the many possible transitions leaving the state. Each transition is labeled with the weight of the transition, which is the product of the relevant model probabilities. At the bottom, the 13 types of state transitions are shown.

state space of the dynamic program by limiting the deviation in cursor positions.

3.2 M-step: updating the parameters

The M-step is standard once we have computed the expected counts of edits in the E-step. For each branch $(k \to l) \in T$ in the phylogeny, we compute the maximum likelihood estimate of the edit parameters $\{\theta_{k\to l}(x \to \beta \,/\, c_1 \,\_\, c_2)\}$. For example, the parameter corresponding to $x =$ /e/, $\beta =$ /e s/, $c_1 =$ ALVEOLAR, $c_2 =$ # is the probability of inserting a final /s/ after an /e/ which is itself preceded by an alveolar phoneme. The probability of each rule is estimated as follows:

$$\theta_{k\to l}(x \to \beta \,/\, c_1 \,\_\, c_2) = \frac{\#(x \to \beta \,/\, c_1 \,\_\, c_2) + \alpha(x \to \beta \,/\, c_1 \,\_\, c_2) - 1}{\sum_{\beta'} \#(x \to \beta' \,/\, c_1 \,\_\, c_2) + \alpha(x \to \beta' \,/\, c_1 \,\_\, c_2) - 1},$$

where α is the concentration hyperparameter of the Dirichlet prior. The value $\alpha - 1$ can be interpreted as the number of pseudocounts for a rule.

4 Experiments

In this section we show the results of our experiments with our model. The experimental conditions are summarized in Table 1, with additional information on the specifics of the experiments presented in Section 4.5. We start with a description of the corpus we created for these experiments.

Experiment	Topology	Heldout
Latin reconstruction (4.2)	1	`la`:293
Italian reconstruction (4.2)	1	`it`:117
Sound changes (4.3)	2	None
Phylogeny selection (4.4)	2, 3, 4	None

Table 1: Conditions under which each of the experiments presented in this section were performed. The *topology* indices correspond to those displayed in Figure 1. Note that by conditional independence, the topology used for Spanish reconstruction reduces to a chain. The heldout column indicates how many words, if any, were heldout for edit distance evaluation, and from which language.

4.1 Corpus

In order to train and evaluate our system, we compiled a corpus of Romance cognate words. The raw data was taken from three sources: the `wiktionary.org` website, a Bible parallel corpus (Resnik et al., 1999) and the Europarl corpus (Koehn, 2002). From an XML dump of the Wiktionary data, we extracted multilingual translations, which provide a list of word tuples in a large number of languages, including a few ancient languages.

The Europarl and the biblical data were processed and aligned in the standard way, using combined GIZA++ alignments (Och and Ney, 2003).

We performed our experiments with four languages from the Romance family (Latin, Italian, Spanish, and Portuguese). For each of these languages, we used a simple in-house rule-based system to convert the words into their IPA representations.[2] After augmenting our alignments with the *transitive closure*[3] of the Europarl, Bible and Wiktionary data, we filtered out non-cognate words by thresholding the ratio of edit distance to word length.[4] The preprocessing is constraining in that we require that all the elements of a tuple to be cognates, which leaves out a significant portion of the data behind (see the row *Full entries* in Table 2). However, our approach relies on this assumption, as there is no explicit model of non-cognate words. An interesting direction for future work is the joint modeling of phonology with the determination of the cognates, but our simpler setting lets us focus on the properties of the edit model. Moreover, the restriction to full entries has the side advantage that the Latin bottleneck prevents the introduction of too many neologisms, which are numerous in the Europarl data, to the final corpus.

Since we used automatic tools for preparing our corpus rather than careful linguistic analysis, our cognate list is much noiser in terms of the presence of borrowed words and phonemeic transcription errors compared to the ones used by previous approaches (Swadesh, 1955; Dyen et al., 1997). The benefit of our mechanical preprocessing is that more cognate data can easily be made available, allowing us to effectively train richer models. We show in the rest of this section that our phonological model can indeed overcome this noise and recover meaningful patterns from the data.

Name	Languages	Tuples	Word forms
Raw sources of data used to create the corpus			
Wiktionary	es,pt,la,it	5840	11724
Bible	la,es	2391	4782
Europarl	es,pt	36905	73773
	it,es	39506	78982
Main stages of preprocessing of the corpus			
Closure	es,pt,la,it	40944	106090
Cognates	es,pt,la,it	27996	69637
Full entries	es,pt,la,it	586	2344

Table 2: Statistics of the dataset we compiled for the evaluation of our model. We show the languages represented, the number of tuples and the number of word forms found in each of the source of data and pre-processing steps involved in the creation of the dataset we used to test our model. By *full entry*, we mean the number of tuples that are jointly considered cognate by our preprocessing system and that have a word form known for each of the languages of interest. These last row forms the dataset used for our experiments.

Language	Baseline	Model	Improvement
Latin	2.84	2.34	9%
Spanish	3.59	3.21	11%

Table 3: Results of the edit distance experiment. The *language* column corresponds to the language held-out for evaluation. We show the mean edit distance across the evaluation examples.

4.2 Reconstruction of word forms

We ran the system using Topology 1 in Figure 1 to demonstrate the the system can propose reasonable reconstructions of Latin word forms on the basis of modern observations. Half of the Latin words at the root of the tree were held out, and the (uniform cost) Levenshtein edit distance from the predicted reconstruction to the truth was computed. Our baseline is to pick randomly, for each heldout node in the tree, an observed neighboring word (i.e. copy one of the modern forms). We stopped EM after 15 iterations, and reported the result on a Viterbi derivation using the parameters obtained. Our model outperformed this baseline by a 9% relative reduction in average edit distance. Similarly, reconstruction of modern forms was also demonstrated, with an improvement of 11% (see Table 3).

To give a qualitative feel for the operation of the system (good and bad), consider the example in Figure 5, taken from this experiment. The Latin *dentis* /dɛntis/ (teeth) is nearly correctly reconstructed as /dɛntes/, reconciling the appearance of the /j/ in the

[2] The tool and the rules we used are available at nlp.cs.berkeley.edu/pages/historical.html.

[3] For example, we would infer from an la-es bible alignment *confessionem-confesión* (confession) and an es-it Europarl alignment *confesión-confessione* that the Latin word *confessionem* and the Italian word *confessione* are related.

[4] To be more precise we keep a tuple $(w_1, w_2, \ldots, w_p)$ iff $\frac{d(w_i,w_j)}{\bar{l}(w_i,w_j)} \leq 0.7$ for all $i, j \in \{1, 2, \ldots, p\}$, where $\bar{l}$ is the mean length $\frac{|w_i|+|w_j|}{2}$ and d is the Levenshtein distance.

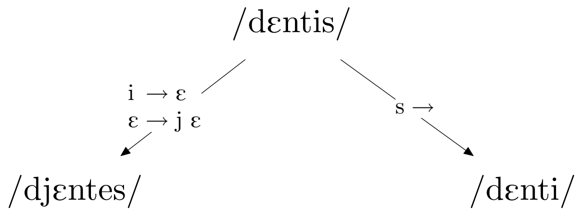


Figure 5: An example of a Latin reconstruction given the Spanish and Italian word forms.

Spanish and the disappearance of the final /s/ in the Italian. Note that the /is/ vs. /es/ ending is difficult to predict in this context (indeed, it was one of the early distinctions to be eroded in vulgar Latin).

While the uniform-cost edit distance misses important aspects of phonology (all phoneme substitutions are not equal, for instance), it is parameter-free and still seems to correlate to a large extent with linguistic quality of reconstruction. It is also superior to held-out log-likelihood, which fails to penalize errors in the modeling assumptions, and to measuring the percentage of perfect reconstructions, which ignores the degree of correctness of each reconstructed word.

4.3 Inference of phonological changes

Another use of our model is to automatically recover the phonological drift processes between known or partially known languages. To facilitate evaluation, we continued in the well-studied Romance evolutionary tree. Again, the root is Latin, but we now add an additional modern language, Portuguese, and two additional hidden nodes. One of the nodes characterizes the least common ancestor of modern Spanish and Portuguese; the other, the least common ancestor of all three modern languages. In Figure 1, Topology 2, these two nodes are labelled `vl` (Vulgar Latin) and `ib` (Proto-Ibero Romance) respectively. Since we are omitting many other branches, these names should not be understood as referring to actual historical proto-languages, but, at best, to collapsed points representing several centuries of evolution. Nonetheless, the major reconstructed rules still correspond to well known phenomena and the learned model generally places them on reasonable branches.

Figure 6 shows the top four general rules for each of the evolutionary branches in this experiment, ranked by the number of times they were used in the derivations during the last iteration of EM. The `la`, `es`, `pt`, and `it` forms are fully observed while the `vl` and `ib` forms are automatically reconstructed. Figure 6 also shows a specific example of the evolution of the Latin *VERBUM* (word/verb), along with the specific edits employed by the model.

While quantitative evaluation such as measuring edit distance is helpful for comparing results, it is also illuminating to consider the plausibility of the learned parameters in a historical light, which we do here briefly. In particular, we consider rules on the branch between `la` and `vl`, for which we have historical evidence. For example, documents such as the *Appendix Probi* (Baehrens, 1922) provide indications of orthographic confusions which resulted from the growing gap between Classical Latin and Vulgar Latin phonology around the 3rd and 4th centuries AD. The *Appendix* lists common misspellings of Latin words, from which phonological changes can be inferred.

On the `la` to `vl` branch, rules for word-final deletion of classical case markers dominate the list (rules ranks 1 and 3 for deletion of final /s/, ranks 2 and 4 for deletion of final /m/). It is indeed likely that these were generally eliminated in Vulgar Latin. For the deletion of the /m/, the *Appendix Probi* contains pairs such as *PASSIM NON PASSI* and *OLIM NON OLI*. For the deletion of final /s/, this was observed in early inscriptions, e.g. *CORNELIO* for *CORNELIOS* (Allen, 1989). The frequent leveling of the distinction between /o/ and /u/ (rules ranked 5 and 6) can be also be found in the *Appendix Probi*: *COLUBER NON COLOBER*. Note that in the specific example shown, the model lowers the orignal /u/ and then re-raises it in the `pt` branch due to a latter process along that branch.

Similarily, major canonical rules were discovered in other branches as well, for example, /v/ to /b/ fortition in Spanish, /s/ to /z/ voicing in Italian, palatalization along several branches, and so on. Of course, the recovered words and rules are not perfect. For example, reconstructed Ibero /trinta/ to Spanish /treinta/ (thirty) is generated in an odd fashion using rules /e/ to /i/ and /n/ to /in/. Moreover, even when otherwise reasonable systematic sound changes are captured, the crudeness of our fixed-granularity contexts can prevent the true context

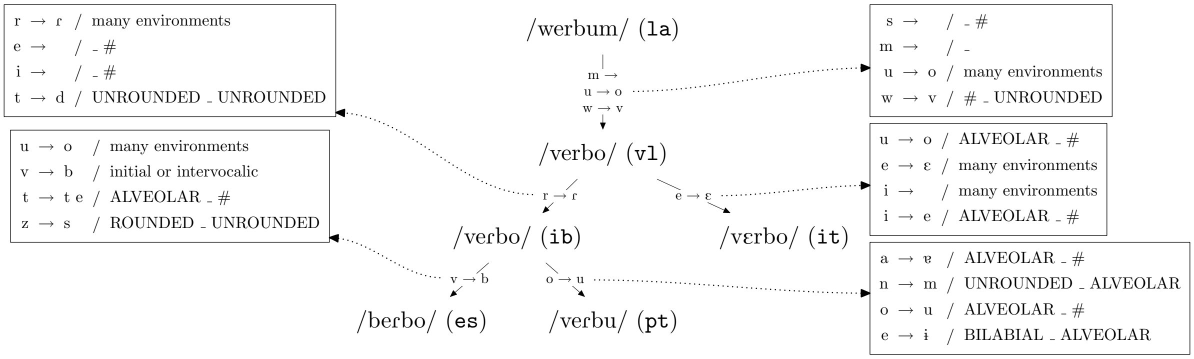


Figure 6: The tree shows the system's hypothesised derivation of a selected Latin word form, *VERBUM* (word/verb) into the modern Spanish, Italian and Portuguese pronunciations. The Latin root and modern leaves were observed while the hidden nodes as well as all the derivations were obtained using the parameters computed by our model after 15 iterations of EM. Nontrivial rules (i.e. rules that are not identities) used at each stage are shown along the corresponding edge. The boxes display the top four nontrivial rules corresponding to each of these evolutionary branches, ordered by the number of time they were applied during the last E round of sampling. Note that since our natural classes are of fixed granularity, some rules must be redundantly discovered, which tends to flood the top of the rule lists with duplicates of the top few rules. We summarized such redundancies in the above tables.

from being captured, resulting in either rules applying with low probability in overly coarse environments or rules being learned redundantly in overly fine environments.

4.4 Selection of phylogenies

In this experiment, we show that our model can be used to select between various topologies of phylogenies. We first presented to the algorithm the universally accepted evolutionary tree corresponding to the evolution of Latin into Spanish, Portuguese and Italian (Topology 2 in Figure 1). We estimated the log-likelihood L^* of the data under this topology. Next, we estimated the log-likelihood L' under two defective topologies (*Topology 3 and *Topology 4). We recorded the log-likelihood ratio $L^* - L'$ after the last iteration of EM. Note that the two likelihoods are comparable since the complexity of the two models is the same.[5]

We obtained a ratio of $L^* - L' = -4458 - (-4766) = 307$ for Topology 2 versus *Topology 3, and $-4877 - (-5125) = 248$ for Topology 2 versus *Topology 4 (the experimental setup is described in Table 1). As one would hope, this log-likelihood ratio is positive in both cases, indicating that the system prefers the true topology over the wrong ones.

While it may seem, at the first glance, that this result is limited in scope, knowing the relative arrangement of all groups of four nodes is actually sufficient for constructing a full-fledged phylogenetic tree. Indeed, *quartet-based* methods, which have been very popular in the computational biology community, are precisely based on this fact (Erdos et al., 1996). There is a rich literature on this subject and approximate algorithms exist which are robust to misclassification of a subset of quartets (Wu et al., 2007).

4.5 More experimental details

This section summarizes the values of the parameters we used in these experiments, their interpretation, and the effect of setting them to other values.

The Dirichlet prior on the parameters can be interpreted as adding pseudocounts to the corresponding edits. It is an important way of infusing parsimony into the model by setting the prior of the self-substitution parameters much higher than that of the other parameters. We used 6.0 as the prior on the self-substitution parameters, and for all environments, 1.1 was divided uniformly across the other edits. As long as the prior on self-substitution is kept within this rough order of magnitude, varying them has a limited effect on our results. We also initialized the parameters with values that encourage self-substitutions. Again, the results were robust to perturbation of initialization as long as the value for self-substitution dominates the other parameters.

The experiments used two natural classes for vowels (rounded and unrounded), and six natural

[5] If a word was not reachable in one of the topology, it was ignored in both models for the computation of the likelihoods.

classes for consonants, based on the place of articulation (alveolar, bilabial, labiodental, palatal, postalveolar, and velar). We conducted experiments to evaluate the effect of using different natural classes and found that finer ones can help if enough data is used for training. We defer the meticulous study of the optimal granularity to future work, as it would be a more interesting experiment under a log-linear model. In such a model, contexts of different granularities can coexist, whereas such coexistence is not recognized by the current model, giving rise to many duplicate rules.

We estimated the bigram phoneme model on the words in the root languages that were not heldout. Just as in machine translation, the language model was found to contribute significantly to reconstruction performance. We tried to increase the weight of the language model by exponentiating it to a power, as is often done in NLP applications, but we did not find that it had any significant impact on performance.

In the reconstruction experiments, when the data was not reachable by the model, the word used in the initialization was used as the prediction, and the evolution of these words were ignored when re-estimating the parameters. Words were initialized by picking at random, for each unobserved node, an observed node's corresponding word.

5 Conclusion

We have presented a novel probabilistic model of diachronic phonology and an associated inference procedure. Our experiments indicate that our model is able to both produce accurate reconstructions as measured by edit distance and identify linguistically plausible rules that account for the phonological changes. We believe that the probabilistic framework we have introduced for diachronic phonology is promising, and scaling it up to richer phylogenetic may indeed reveal something insightful about language change.

6 Acknowledgement

We would like to thank Bonnie Chantarotwong for her help with the IPA converter and our reviewers for their comments. This work was supported by a FQRNT fellowship to the first author, a NDSEG fellowship to the second author, NSF grant number BCS-0631518 to the third author, and a Microsoft Research New Faculty Fellowship to the fourth author.

References

W. Sidney Allen. 1989. *Vox Latina: The Pronunciation of Classical Latin*. Cambridge University Press.

W.A. Baehrens. 1922. *Sprachlicher Kommentar zur vulgärlateinischen Appendix Probi*. Halle (Saale) M. Niemeyer.

L. Campbell. 1998. *Historical Linguistics*. The MIT Press.

N. Chomsky and M. Halle. 1968. *The Sound Pattern of English*. Harper & Row.

I. Dyen, J.B. Kruskal, and P. Black. 1997. FILE IE-DATA1. Available at http://www.ntu.edu.au/education/langs/ielex/IE-DATA1.

P. L. Erdos, M. A. Steel, L. A. Szekely, and T. J. Warnow. 1996. Local quartet splits of a binary tree infer all quartet splits via one dyadic inference rule. Technical report, DIMACS.

S. N. Evans, D. Ringe, and T. Warnow. 2004. Inference of divergence times as a statistical inverse problem. In P. Forster and C. Renfrew, editors, *Phylogenetic Methods and the Prehistory of Languages*. McDonald Institute Monographs.

Joseph Felsenstein. 2003. *Inferring Phylogenies*. Sinauer Associates.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

R. D. Gray and Q. Atkinson. 2003. Language-tree divergence times support the Anatolian theory of Indo-European origins. *Nature*.

John P. Huelsenbeck, Fredrik Ronquist, Rasmus Nielsen, and Jonathan P. Bollback. 2001. Bayesian inference of phylogeny and its impact on evolutionary biology. *Science*.

P. Koehn. 2002. Europarl: A Multilingual Corpus for Evaluation of Machine Translation.

G. Kondrak. 2002. *Algorithms for Language Reconstruction*. Ph.D. thesis, University of Toronto.

S. Li, D. K. Pearl, and H. Doss. 2000. Phylogenetic tree construction using Markov chain Monte Carlo. *Journal of the American Statistical Association*.

Bob Mau and M.A. Newton. 1997. Phylogenetic inference for binary data on dendrograms using markov chain monte carlo. *Journal of Computational and Graphical Statistics*.

L. Nakhleh, D. Ringe, and T. Warnow. 2005. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language*, 81:382–420.

F. J. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29:19–51.

P. Resnik, Mari Broman Olsen, and Mona Diab. 1999. The bible as a parallel corpus: Annotating the "book of 2000 tongues". *Computers and the Humanities*, 33(1-2):129–153.

D. Ringe, T. Warnow, and A. Taylor. 2002. Indoeuropean and computational cladistics. *Transactions of the Philological Society*, 100:59–129.

M. Swadesh. 1955. Towards greater accuracy in lexicostatistic dating. *Journal of American Linguistics*, 21:121–137.

A. Venkataraman, J. Newman, and J.D. Patrick. 1997. A complexity measure for diachronic chinese phonology. In J. Coleman, editor, *Computational Phonology*. Association for Computational Linguistics.

G. Wu, J. A. You, and G. Lin. 2007. Quartet-based phylogeny reconstruction with answer set programming. *IEEE/ACM Transactions on computational biology*, 4:139–152.

Ziheng Yang and Bruce Rannala. 1997. Bayesian phylogenetic inference using dna sequences: A markov chain monte carlo method. *Molecular Biology and Evolution 14*.

Learning Structured Models for Phone Recognition

Slav Petrov **Adam Pauls** **Dan Klein**
Computer Science Department, EECS Divison
University of California at Berkeley
Berkeley, CA, 94720, USA
{petrov,adpauls,klein}@cs.berkeley.edu

Abstract

We present a maximally streamlined approach to learning HMM-based acoustic models for automatic speech recognition. In our approach, an initial monophone HMM is iteratively refined using a split-merge EM procedure which makes no assumptions about subphone structure or context-dependent structure, and which uses only a single Gaussian per HMM state. Despite the much simplified training process, our acoustic model achieves state-of-the-art results on phone classification (where it outperforms almost all other methods) and competitive performance on phone recognition (where it outperforms standard CD triphone / subphone / GMM approaches). We also present an analysis of what is and is not learned by our system.

1 Introduction

Continuous density hidden Markov models (HMMs) underlie most automatic speech recognition (ASR) systems in some form. While the basic algorithms for HMM learning and inference are quite general, acoustic models of speech standardly employ rich speech-specific structures to improve performance. For example, it is well known that a *monophone* HMM with one state per phone is too coarse an approximation to the true articulatory and acoustic process. The HMM state space is therefore refined in several ways. To model phone-internal dynamics, phones are split into *beginning*, *middle*, and *end* subphones (Jelinek, 1976). To model cross-phone coarticulation, the states of the HMM are refined by splitting the phones into context-dependent *triphones*. These states are then re-clustered (Odell, 1995) and the parameters of their observation distributions are tied back together (Young and Woodland, 1994). Finally, to model complex emission densities, states emit mixtures of multivariate Gaussians. This standard structure is shown schematically in Figure 1. While this rich structure is phonetically well-motivated and empirically successful, so much structural bias may be unnecessary, or even harmful. For example in the domain of syntactic parsing with probabilistic context-free grammars (PCFGs), a surprising recent result is that automatically induced grammar refinements can outperform sophisticated methods which exploit substantial manually articulated structure (Petrov et al., 2006).

In this paper, we consider a much more automatic, data-driven approach to learning HMM structure for acoustic modeling, analagous to the approach taken by Petrov et al. (2006) for learning PCFGs. We start with a minimal monophone HMM in which there is a single state for each (context-independent) phone. Moreover, the emission model for each state is a single multivariate Gaussian (over the standard MFCC acoustic features). We then iteratively refine this minimal HMM through state splitting, adding complexity as needed. States in the refined HMMs are always substates of the original HMM and are therefore each identified with a unique base phone. States are split, estimated, and (perhaps) merged, based on a likelihood criterion. Our model never allows explicit Gaussian mixtures, though substates may develop similar distributions and thereby emulate such mixtures.

In principle, discarding the traditional structure can either help or hurt the model. Incorrect prior splits can needlessly fragment training data and incorrect prior tying can limit the model's expressivity. On the other hand, correct assumptions can increase the efficiency of the learner. Empirically,

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 897–905, Prague, June 2007.

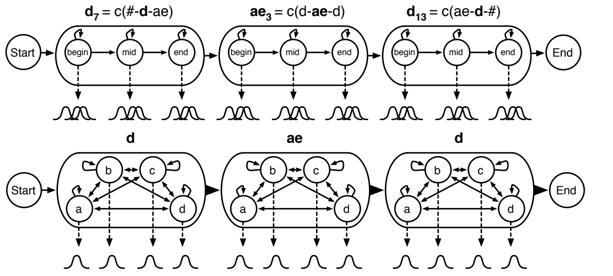


Figure 1: Comparison of the standard model to our model (here shown with $k = 4$ subphones per phone) for the word *dad*. The dependence of subphones across phones in our model is not shown, while the context clustering in the standard model is shown only schematically.

we show that our automatic approach outperforms classic systems on the task of phone recognition on the TIMIT data set. In particular, it outperforms standard state-tied triphone models like Young and Woodland (1994), achieving a phone error rate of 26.4% versus 27.7%. In addition, our approach gives state-of-the-art performance on the task of phone classification on the TIMIT data set, suggesting that our learned structure is particularly effective at modeling phone-internal structure. Indeed, our error rate of 21.4% is outperformed only by the recent structured margin approach of Sha and Saul (2006). It remains to be seen whether these positive results on acoustic modeling will facilitate better word recognition rates in a large vocabulary speech recognition system.

We also consider the structures learned by the model. Subphone structure is learned, similar to, but richer than, standard begin-middle-end structures. Cross-phone coarticulation is also learned, with classic phonological classes often emerging naturally.

Many aspects of this work are intended to simplify rather than further articulate the acoustic process. It should therefore be clear that the basic techniques of splitting, merging, and learning using EM are not in themselves new for ASR. Nor is the basic latent induction method new (Matsuzaki et al., 2005; Petrov et al., 2006). What is novel in this paper is (1) the construction of an automatic system for acoustic modeling, with substantially streamlined structure, (2) the investigation of variational inference for such a task, (3) the analysis of the kinds of structures learned by such a system, and (4) the empirical demonstration that such a system is not only competitive with the traditional approach, but can indeed outperform even very recent work on some preliminary measures.

2 Learning

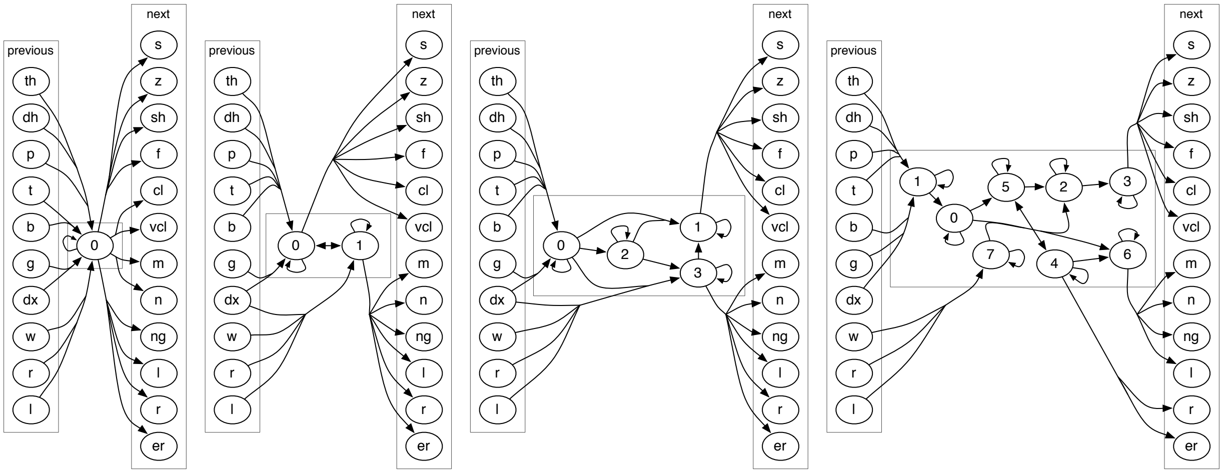
In the following, we propose a greatly simplified model that does not impose any manually specified structural constraints. Instead of specifying structure *a priori*, we use the Expectation-Maximization (EM) algorithm for HMMs (Baum-Welch) to automatically induce the structure in a way that maximizes data likelihood.

In general, our training data consists of sets of acoustic observation sequences and phone level transcriptions $\mathbf{r}$ which specify a sequence of phones from a set of phones Y, but does not label each time frame with a phone. We refer to an observation sequence as $\mathbf{x} = x_1, \ldots, x_T$ where $x_i \in \mathbb{R}^{39}$ are standard MFCC features (Davis and Mermelstein, 1980). We wish to induce an HMM over a set of states S for which we also have a function $\pi : S \rightarrow Y$ that maps every state in S to a phone in Y. Note that in the usual formulation of the EM algorithm for HMMs, one is interested in learning HMM parameters θ that maximize the likelihood of the observations $\mathrm{P}(\mathbf{x}|\theta)$; in contrast, we aim to maximize the joint probability of our observations and phone transcriptions $\mathrm{P}(\mathbf{x}, \mathbf{r}|\theta)$ or observations and phone sequences $\mathrm{P}(\mathbf{x}, \mathbf{y}|\theta)$ (see below). We now describe this relatively straightforward modification of the EM algorithm.

2.1 The Hand-Aligned Case

For clarity of exposition we first consider a simplified scenario in which we are given hand-aligned phone labels $\mathbf{y} = y_1, \ldots, y_T$ for each time t, as is the case for the TIMIT dataset. Our procedure does not require such extensive annotation of the training data and in fact gives better performance when the exact transition point between phones are not prespecified but learned.

We define forward and backward probabilities (Rabiner, 1989) in the following way: the forward probability is the probability of observing the sequence $x_1, \ldots, x_t$ with transcription $y_1, \ldots, y_t$ and



(a) (b) (c) (d)

Figure 2: Iterative refinement of the /ih/ phone with $1, 2, 4, 8$ substates.

ending in state s at time t:

$$\alpha_t(s) = \mathrm{P}(x_1, \ldots, x_t, y_1, \ldots y_t, s_t = s|\lambda),$$

and the backward probability is the probability of observing the sequence $x_{t+1}, \ldots, x_T$ with transcription $y_{t+1}, \ldots, y_T$, given that we start in state s at time t:

$$\beta_t(s) = \mathrm{P}(x_{t+1}, \ldots, x_T, y_{t+1}, \ldots, y_T|s_t = s, \lambda),$$

where λ are the model parameters. As usual, we parameterize our HMMs with $a_{ss'}$, the probability of transitioning from state s to s', and $b_s(x) \sim \mathcal{N}(\mu_s, \Sigma_s)$, the probability emitting the observation x when in state s.

These probabilities can be computed using the standard forward and backward recursions (Rabiner, 1989), except that at each time t, we only consider states s_t for which $\pi(s_t) = y_t$, because we have hand-aligned labels for the observations. These quantities also allow us to compute the posterior counts necessary for the E-step of the EM algorithm.

2.2 Splitting

One way of inducing arbitrary structural annotations would be to split each HMM state in into m substates, and re-estimate the parameters for the split HMM using EM. This approach has two major drawbacks: for larger m it is likely to converge to poor local optima, and it allocates substates uniformly across all states, regardless of how much annotation is required for good performance.

To avoid these problems, we apply a hierarchical parameter estimation strategy similar in spirit to the work of Sankar (1998) and Ueda et al. (2000), but here applied to HMMs rather than to GMMs. Beginning with the baseline model, where each state corresponds to one phone, we repeatedly split and re-train the HMM. This strategy ensures that each split HMM is initialized “close” to some reasonable maximum.

Concretely, each state s in the HMM is split in two new states s_1, s_2 with $\pi(s_1) = \pi(s_2) = \pi(s)$. We initialize EM with the parameters of the previous HMM, splitting every previous state s in two and adding a small amount of randomness $\epsilon \leq 1\%$ to its transition and emission probabilities to break symmetry:

$$a_{s_1 s'} \propto a_{ss'} + \epsilon,$$

$$b_{s_1}(o) \sim \mathcal{N}(\mu_s + \epsilon, \Sigma_s),$$

and similarly for s_2. The incoming transitions are split evenly.

We then apply the EM algorithm described above to re-estimate these parameters before performing subsequent split operations.

2.3 Merging

Since adding substates divides HMM statistics into many bins, the HMM parameters are effectively estimated from less data, which can lead to overfitting. Therefore, it would be to our advantage to split substates only where needed, rather than splitting them all.

We realize this goal by merging back those splits $s \rightarrow s_1 s_2$ for which, if the split were reversed, the loss in data likelihood would be smallest. We approximate the loss in data likelihood for a merge $s_1 s_2 \rightarrow s$ with the following likelihood ratio (Petrov et al., 2006):

$$\Delta(s_1\ s_2 \rightarrow s) = \prod_{sequences} \prod_t \frac{\mathrm{P}^t(\mathbf{x}, \mathbf{y})}{\mathrm{P}(\mathbf{x}, \mathbf{y})}.$$

Here $\mathrm{P}(\mathbf{x}, \mathbf{y})$ is the joint likelihood of an emission sequence $\mathbf{x}$ and associated state sequence $\mathbf{y}$. This quantity can be recovered from the forward and backward probabilities using

$$\mathrm{P}(\mathbf{x}, \mathbf{y}) = \sum_{s:\pi(s)=y_t} \alpha_t(s) \cdot \beta_t(s).$$

$\mathrm{P}^t(\mathbf{x}, \mathbf{y})$ is an approximation to the same joint likelihood where states s_1 and s_2 are merged. We approximate the true loss by only considering merging states s_1 and s_2 at time t, a value which can be efficiently computed from the forward and backward probabilities. The forward score for the merged state s at time t is just the sum of the two split scores:

$$\hat{\alpha}_t(s) = \alpha_t(s_1) + \alpha_t(s_2),$$

while the backward score is a weighted sum of the split scores:

$$\hat{\beta}_t(s) = p_1 \beta_t(s_1) + p_2 \beta_t(s_2),$$

where p_1 and p_2 are the relative (posterior) frequencies of the states s_1 and s_2.

Thus, the likelihood after merging s_1 and s_2 at time t can be computed from these merged forward and backward scores as:

$$P^t(\mathbf{x}, \mathbf{y}) = \hat{\alpha}_t(s) \cdot \hat{\beta}_t(s) + \sum_{s'} \alpha_t(s') \cdot \beta_t(s')$$

where the second sum is over the other substates of x_t, i.e. $\{s' : \pi(s') = x_t, s' \notin \{s_1, s_2\}\}$. This expression is an approximation because it neglects interactions between instances of the same states at multiple places in the same sequence. In particular, since phones frequently occur with multiple consecutive repetitions, this criterion may vastly overestimate the actual likelihood loss. As such, we also implemented the exact criterion, that is, for each split, we formed a new HMM with s_1 and s_2 merged and calculated the total data likelihood. This method is much more computationally expensive, requiring a full forward-backward pass through the data for each potential merge, and was not found to produce noticeably better performance. Therefore, all experiments use the approximate criterion.

2.4 The Automatically-Aligned Case

It is straightforward to generalize the hand-aligned case to the case where the phone transcription is known, but no frame level labeling is available. The main difference is that the phone boundaries are not known in advance, which means that there is now additional uncertainty over the phone states. The forward and backward recursions must thus be expanded to consider all state sequences that yield the given phone transcription. We can accomplish this with standard Baum-Welch training.

3 Inference

An HMM over refined subphone states $s \in S$ naturally gives posterior distributions $\mathrm{P}(\mathbf{s}|\mathbf{x})$ over *sequences* of states $\mathbf{s}$. We would ideally like to extract the transcription $\mathbf{r}$ of underlying phones which is most probable according to this posterior[1]. The transcription is two stages removed from $\mathbf{s}$. First, it collapses the distinctions between states s which correspond to the same phone $y = \pi(s)$. Second, it collapses the distinctions between where phone transitions exactly occur. Viterbi state sequences can easily be extracted using the basic Viterbi algorithm. On the other hand, finding the best phone sequence or transcription is intractable.

As a compromise, we extract the phone sequence (not transcription) which has highest probability in a variational approximation to the true distribution (Jordan et al., 1999). Let the true posterior distribution over phone sequences be $\mathrm{P}(\mathbf{y}|\mathbf{x})$. We form an approximation $\mathrm{Q}(\mathbf{y}) \approx \mathrm{P}(\mathbf{y}|\mathbf{x})$, where Q is an approximation specific to the sequence $\mathbf{x}$ and factor-

[1] Remember that by "transcription" we mean a sequence of phones with duplicates removed.

izes as:

$$\mathrm{Q}(\mathbf{y}) = \prod_t q(t, x_t, y_{t+1}).$$

We would like to fit the values q, one for each time step and state-state pair, so as to make Q as close to P as possible:

$$\min_q KL(\mathrm{P}(\mathbf{y}|\mathbf{x})||\mathrm{Q}(\mathbf{y})).$$

The solution can be found analytically using Lagrange multipliers:

$$q(t, y, y') = \frac{\mathrm{P}(Y_t = y, Y_{t+1} = y'|\mathbf{x})}{\mathrm{P}(Y_t = y|\mathbf{x})}.$$

where we have made the position-specific random variables Y_t explicit for clarity. This approximation depends only on our ability to calculate posteriors over phones or phone-phone pairs at individual positions t, which is easy to obtain from the state posteriors, for example:

$$\mathrm{P}(Y_t = y, Y_{t+1} = y'|\mathbf{x}) = \frac{\displaystyle\sum_{s:\pi(s)=y} \sum_{s':\pi(s')=y'} \alpha_t(s) a_{ss'} b_{s'}(x_t) \beta_{t+1}(s')}{\mathrm{P}(\mathbf{x})}$$

Finding the Viterbi *phone* sequence in the approximate distribution Q, can be done with the Forward-Backward algorithm over the lattice of q values.

4 Experiments

We tested our model on the TIMIT database, using the standard setups for phone recognition and phone classification. We partitioned the TIMIT data into training, development, and (core) test sets according to standard practice (Lee and Hon, 1989; Gunawardana et al., 2005; Sha and Saul, 2006). In particular, we excluded all *sa* sentences and mapped the 61 phonetic labels in TIMIT down to 48 classes before training our HMMs. At evaluation, these 48 classes were further mapped down to 39 classes, again in the standard way.

MFCC coefficients were extracted from the TIMIT source as in Sha and Saul (2006), including delta and delta-delta components. For all experiments, our system and all baselines we implemented used *full covariance* when parameterizing emission models.[2] All Gaussians were endowed with weak inverse Wishart priors with zero mean and identity covariance.[3]

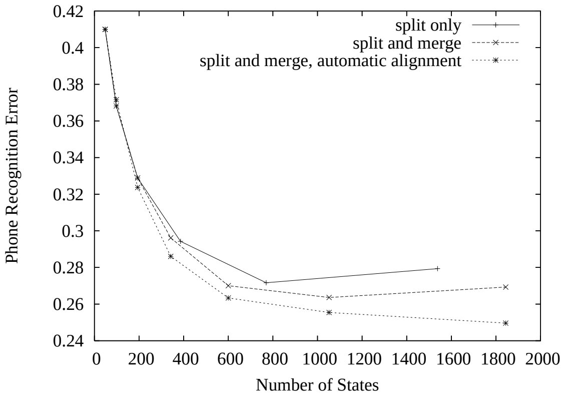


Figure 3: Phone recognition error for models of increasing size

4.1 Phone Recognition

In the task of phone recognition, we fit an HMM whose output, with subsequent states collapsed, corresponds to the training transcriptions. In the TIMIT data set, each frame is manually phone-annotated, so the only uncertainty in the basic setup is the identity of the (sub)states at each frame.

We therefore began with a single state for each phone, in a fully connected HMM (except for special treatment of dedicated start and end states). We incrementally trained our model as described in Section 2, with up to 6 split-merge rounds. We found that reversing 25% of the splits yielded good overall performance while maintaining compactness of the model.

We decoded using the variational decoder described in Section 3. The output was then scored against the reference phone transcription using the standard string edit distance.

During both training and decoding, we used “flattened” emission probabilities by exponentiating to some $0 < \gamma < 1$. We found the best setting for γ to be 0.2, as determined by tuning on the development set. This flattening compensates for the non-

[2] Most of our findings also hold for diagonal covariance Gaussians, albeit the final error rates are 2-3% higher.

[3] Following previous work with PCFGs (Petrov et al., 2006), we experimented with smoothing the substates towards each other to prevent overfitting, but we were unable to achieve any performance gains.

Method	Error Rate
State-Tied Triphone HMM (Young and Woodland, 1994)	27.7%[1]
Gender Dependent Triphone HMM (Lamel and Gauvain, 1993)	27.1%[1]
This Paper	**26.4%**
Bayesian Triphone HMM (Ming and Smith, 1998)	25.6%
Heterogeneous classifiers (Halberstadt and Glass, 1998)	24.4%

Table 1: Phone recognition error rates on the TIMIT core test from Glass (2003).

[1] These results are on a slightly easier test set.

independence of the frames, partially due to overlapping source samples and partially due to other unmodeled correlations.

Figure 3 shows the recognition error as the model grows in size. In addition to the basic setup described so far (*split and merge*), we also show a model in which merging was not performed (*split only*). As can be seen, the merging phase not only decreases the number of HMM states at each round, but also improves phone recognition error at each round.

We also compared our hierarchical *split only* model with a model where we directly split all states into 2^k substates, so that these models had the same number of states as a a hierarchical model after k split and merge cycles. While for small k, the difference was negligible, we found that the error increased by 1% absolute for $k = 5$. This trend is to be expected, as the possible interactions between the substates grows with the number of substates.

Also shown in Figure 3, and perhaps unsurprising, is that the error rate can be further reduced by allowing the phone boundaries to drift from the manual alignments provided in the TIMIT training data. The *split and merge, automatic alignment* line shows the result of allowing the EM fitting phase to reposition each phone boundary, giving absolute improvements of up to 0.6%.

We investigated how much improvement in accuracy one can gain by computing the variational approximation introduced in Section 3 versus extracting the Viterbi state sequence and projecting that sequence to its phone transcription. The gap varies, but on a model with roughly 1000 states (5 split-merge rounds), the variational decoder decreases error from 26.5% to 25.6%. The gain in accuracy comes at a cost in time: we must run a (possibly pruned) Forward-Backward pass over the full state space S, then another over the smaller phone space Y. In our experiments, the cost of variational decoding was a factor of about 3, which may or may not justify a relative error reduction of around 4%.

Method	Error Rate
GMM Baseline (Sha and Saul, 2006)	26.0%
HMM Baseline (Gunawardana et al., 2005)	25.1%
SVM (Clarkson and Moreno, 1999)	22.4%
Hidden CRF (Gunawardana et al., 2005)	21.7%
This Paper	**21.4%**
Large Margin GMM (Sha and Saul, 2006)	21.1%

Table 2: Phone classification error rates on the TIMIT core test.

The performance of our best model (split and merge, automatic alignment, and variational decoding) on the test set is 26.4%. A comparison of our performance with other methods in the literature is shown in Table 1. Despite our structural simplicity, we outperform state-tied triphone systems like Young and Woodland (1994), a standard baseline for this task, by nearly 2% absolute. However, we fall short of the best current systems.

4.2 Phone Classification

Phone classification is the fairly constrained task of classifying in isolation a sequence of frames which is known to span exactly one phone. In order to quantify how much of our gains over the triphone baseline stem from modeling context-dependencies and how much from modeling the inner structure of the phones, we fit separate HMM models for each phone, using the same split and merge procedure as above (though in this case only manual alignments are reasonable because we test on manual segmentations). For each test frame sequence, we compute the likelihood of the sequence from the forward probabilities of each individual phone HMM. The phone giving highest likelihood to the input was selected. The error rate is a simple fraction of test phones classified correctly.

Table 2 shows a comparison of our performance with that of some other methods in the literature. A minimal comparison is to a GMM with the same number of mixtures per phone as our model's maxi-

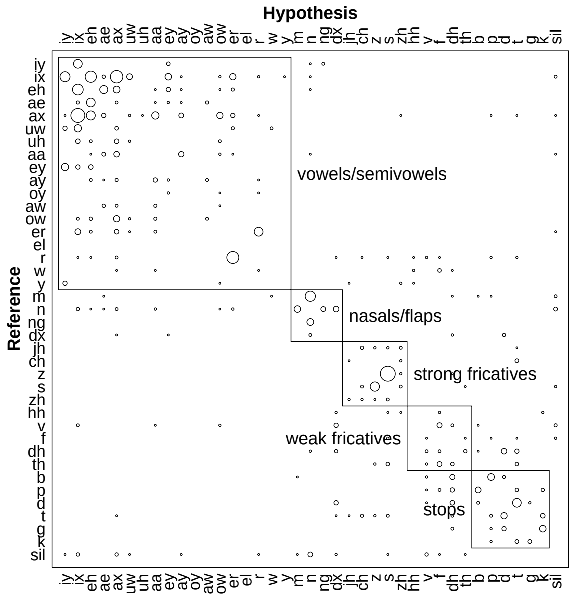


Figure 4: Phone confusion matrix. 76% of the substitutions fall within the shown classes.

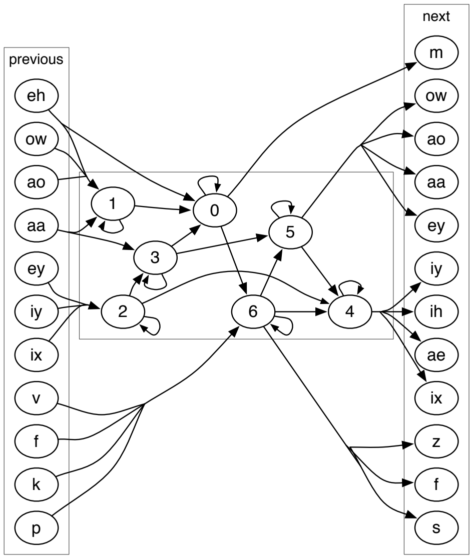


Figure 5: Phone contexts and subphone structure. The /l/ phone after 3 split-merge iterations is shown.

mum substates per phone. While these models have the same number of total Gaussians, in our model the Gaussians are correlated temporally, while in the GMM they are independent. Enforcing begin-middle-end HMM structure (see *HMM Baseline*) increases accuracy somewhat, but our more general model clearly makes better use of the available parameters than those baselines.

Indeed, our best model achieves a surprising performance of 21.4%, greatly outperforming other generative methods and achieving performance competitive with state-of-the-art discriminative methods. Only the recent structured margin approach of Sha and Saul (2006) gives a better performance than our model. The strength of our system on the classification task suggests that perhaps it is modeling phone-internal structure more effectively than cross-phone context.

5 Analysis

While the overall phone recognition and classification numbers suggest that our system is broadly comparable to and perhaps in certain ways superior to classical approaches, it is illuminating to investigate what is and is not learned by the model.

Figure 4 gives a confusion matrix over the substitution errors made by our model. The majority of the confusions are within natural classes. Some particularly frequent and reasonable confusions arise between the consonantal /r/ and the vocalic /er/ (the same confusion arises between /l/ and /el/, but the standard evaluation already collapses this distinction), the reduced vowels /ax/ and /ix/, the voiced and voiceless alveolar sibilants /z/ and /s/, and the voiced and voiceless stop pairs. Other vocalic confusions are generally between vowels and their corresponding reduced forms. Overall, 76% of the substitutions are within the broad classes shown in the figure.

We can also examine the substructure learned for the various phones. Figure 2 shows the evolution of the phone /ih/ from a single state to 8 substates during split/merge (no merges were chosen for this phone), using hand-alignment of phones to frames. These figures were simplified from the complete state transition matrices as follows: (1) adjacent phones' substates are collapsed, (2) adjacent phones are selected based on frequency and inbound probability (and forced to be the same across figures), (3) infrequent arcs are suppressed. In the first split, (b), a sonorant / non-sonorant distinction is learned over adjacent phones, along with a state chain which captures basic duration (a self-looping state gives an exponential model of duration; the sum of two such states is more expressive). Note that the natural classes interact with the chain in a way which allows duration to depend on context. In further refinements, more structure is added, including a two-track path in (d) where one track captures the distinct effects on higher formants of r-coloring and nasalization. Figure 5 shows the corresponding diagram for /l/, where some merging has also occurred. Different natural classes emerge in this case, with, for example, preceding states partitioned into front/high vowels vs. rounded vowels vs. other vowels vs. consonants. Following states show a front/back distinction and a consonant distinction, and the phone /m/ is treated specially, largely because the /lm/ sequence tends to shorten the /l/ substantially. Note again how context, internal structure, and duration are simultaneously modeled. Of course, it should be emphasized that *post hoc* analysis of such structure is a simplification and prone to seeing what one expects; we present these examples to illustrate the broad kinds of patterns which are detected.

As a final illustration of the nature of the learned models, Table 3 shows the number of substates allocated to each phone by the split/merge process (the maximum is 32 for this stage) for the case of hand-aligned (left) as well as automatically-aligned (right) phone boundaries. Interestingly, in the hand-aligned case, the vowels absorb most of the complexity since many consonantal cues are heavily evidenced on adjacent vowels. However, in the automatically-aligned case, many vowel frames with substantial consontant coloring are re-allocated to those adjacent consonants, giving more complex consonants, but comparatively less complex vowels.

6 Conclusions

We have presented a minimalist, automatic approach for building an accurate acoustic model for phonetic classification and recognition. Our model does not require any *a priori* phonetic bias or manual specification of structure, but rather induces the structure in an automatic and streamlined fashion. Starting from a minimal monophone HMM, we automatically learn models that achieve highly competitive performance. On the TIMIT phone recognition task our model clearly outperforms standard state-tied triphone models like Young and Woodland (1994). For phone classification, our model achieves performance competitive with the state-of-the-art discriminative methods (Sha and Saul, 2006), despite being generative in nature. This result together with our analysis of the context-dependencies and substructures that are being learned, suggests that our model is particularly well suited for modeling phone-internal structure. It does, of course remain to be seen if and how these benefits can be scaled to larger systems.

Vowels		
aa	31	32
ae	32	17
ah	31	8
ao	32	23
aw	18	6
ax	18	3
ay	32	28
eh	32	16
el	6	4
en	4	3
er	32	31
ey	32	30
ih	32	11
ix	31	16
iy	31	32
ow	26	10
oy	4	4
uh	5	2
uw	21	8
Consonants		
b	2	32
ch	13	30
d	2	14
dh	6	31
dx	2	3
f	32	32
g	2	15
hh	3	5
jh	3	16
k	30	32
l	25	32
m	25	25
n	29	32
ng	3	4
p	5	24
r	32	32
s	32	32
sh	30	32
t	24	32
th	8	11
v	23	11
w	10	21
y	3	7
z	31	32
zh	2	2
Other		
epi	2	4
sil	32	32
vcl	29	30
cl	31	32

Table 3: Number of substates allocated per phone. The left column gives the number of substates allocated when training on manually aligned training sequences, while the right column gives the number allocated when we automatically determine phone boundaries.

References

P. Clarkson and P. Moreno. 1999. On the use of Support Vector Machines for phonetic classification. In *ICASSP '99*.

S. B. Davis and P. Mermelstein. 1980. Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4).

J. Glass. 2003. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17(2).

A. Gunawardana, M. Mahajan, A. Acero, and J. Platt. 2005. Hidden Conditional Random Fields for phone recognition. In *Eurospeech '05*.

A. K. Halberstadt and J. R. Glass. 1998. Heterogeneous measurements and multiple classifiers for speech recognition. In *ICSLP '98*.

F. Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE.*

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. 1999. An introduction to variational methods for graphical models. *Learning in Graphical Models*.

L. Lamel and J. Gauvain. 1993. Cross-lingual experiments with phone recognition. In *ICASSP '93*.

K. F. Lee and H. W. Hon. 1989. Speaker-independent phone recognition using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11).

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL '05*.

J. Ming and F.J. Smith. 1998. Improved phone recognition using Bayesian triphone models. In *ICASSP '98*.

J. J. Odell. 1995. *The Use of Context in Large Vocabulary Speech Recognition*. Ph.D. thesis, University of Cambridge.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL '06*.

L. Rabiner. 1989. A Tutorial on hidden Markov models and selected applications in speech recognition. In *IEEE*.

A. Sankar. 1998. Experiments with a Gaussian merging-splitting algorithm for HMM training for speech recognition. In *DARPA Speech Recognition Workshop '98*.

F. Sha and L. K. Saul. 2006. Large margin Gaussian mixture modeling for phonetic classification and recognition. In *ICASSP '06*.

N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. 2000. Split and Merge EM algorithm for mixture models. *Neural Computation*, 12(9).

S. J. Young and P. C. Woodland. 1994. State clustering in HMM-based continuous speech recognition. *Computer Speech and Language*, 8(4).

Inducing Search Keys for Name Filtering

L. Karl Branting
The MITRE Corporation
7467 Ridge Road, Suite 140
Hanover, MD 21076, U.S.A.
lbranting@mitre.org

Abstract

This paper describes ETK (Ensemble of Transformation-based Keys) a new algorithm for inducing search keys for name filtering. ETK has the low computational cost and ability to filter by phonetic similarity characteristic of phonetic keys such as Soundex, but is adaptable to alternative similarity models. The accuracy of ETK in a preliminary empirical evaluation suggests that it is well-suited for phonetic filtering applications such as recognizing alternative cross-lingual transliterations.

1 Introduction

The task of *name matching*—recognizing when two orthographically distinct strings are likely to denote the same individual—occurs in a wide variety of important applications, including law enforcement, national security, and maintenance of government and commercial records. Coreference resolution, speech understanding, and detection of aliases and duplicate names all require name matching.

The orthographic variations that give rise to the name-matching task can result from a variety of factors, including transcription and OCR errors, and spelling variations. In many applications, cross-lingual transliterations are a particularly important source of variation. For example, romanized Arabic names are phonetic transcriptions of sounds that have no direct equivalent in English, *e.g.*, “Mohamed” or “Muhammet” are two of many possible transliterations for the same Arabic name.

Name matching can be viewed as a type of range query in which the input is a set of patterns (such as names on an immigration-control watch list), a collection of text strings (such as a passenger list), a distance metric for calculating the degree of relevant dissimilarity between pairs of strings, and a match threshold expressing the maximum allowable distance between matching names. The goal is to find all text/pattern pairs whose distance under the metric is less than or equal to the threshold. In the simplest case, patterns and the text strings with which they are matched are both individual words. In the general case, the text may not be segmented into strings corresponding to possible names.

Distance metrics for name matching are typically computationally expensive. For example, determining the edit distance between two strings of length n and m requires, in the general case, nm steps. Metrics based on algorithms that learn from examples of strings that should match (Bilenko et al., 2003; Ristad and Yianilos, 1998) and metrics that use phonetic similarity criterion, e.g., (Kondrak, 2000) are no less expensive than edit distance.

The computational expense of distance metrics means that tractable name matching on large texts typically requires an inexpensive, high-recall *filtering* step to find a subset of the original text to which the expensive similarity metric will be applied. Desiderata for filtering include the following:

1. **High recall.** The recall of the entire name-matching process is bounded by the recall of the filtering step, so high filtering recall is essential.

2. **Efficiency.** Filtering is useful only to the extend that it requires less computational expense than applying the similarity metric to each pattern/text pair. The computational expense of

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 906–914, Prague, June 2007. ©2007 Association for Computational Linguistics

a filtering algorithm itself must therefore be less than the cost of the metric calls eliminated through the filtering process. Typically, the cost of the metric is so much higher than the filtering cost that the latter can be neglected. Under these circumstances, precision is a satisfactory proxy for efficiency.

3. **Adaptability to specific distance metrics.** High precision and recall are achievable in filtering only if the filtering criterion corresponds to the distance metric. For example, if a distance metric is based on phonetic differences between strings, a filtering algorithm that selects candidate text strings based on orthographic differences may perform poorly. Similarly, poor performance may result from use of a filtering algorithm based on phonetic differences if the distance metric is based on orthographic differences. For example, "LAYTON" and "LEIGHTON" differ by a large edit distance but are phonetically identical (in most dialects), whereas "BOUGH" and "ROUGH" are orthographically similar but phonetically dissimilar. An ideal filtering algorithm should be adaptable to any particular distance metric.

This paper describes ETK (Ensemble of Transformation-based Keys) a new algorithm for inducing filters that satisfy the three criteria above. ETK is similar to phonetic search key algorithms such as Soundex and shares phonetic search key algorithms' low computational expense and ability to filter by phonetic similarity. However, ETK has the advantage that it is adaptable to alternative distance metrics and is therefore applicable to a wider range of circumstances than static key algorithms.

The next section describes previous work in name filtering. Section 3 describes the ETK algorithm in detail, and a preliminary evaluation on English and German surnames is set forth in Section 4.

2 Previous Work

The division of the retrieval task into an inexpensive, high-recall filtering stage followed by a more expensive high-precision stage emerged independently in a variety of different areas of computer science. This approach is termed *two-stage retrieval* in the Information Retrieval literature (Shin and Zhang, 1998), *MAC/FAC* by some researchers in analogy (Gentner and Forbus, 1991), *blocking* in the statistical record linkage literature (Cohen et al., 2003), and *filtering* in the approximate string matching literature (Navarro, 2001).

The two most common approaches to filtering that have been applied to name matching are indexing by *phonetic keys* and indexing by *ngrams*. Two less well known filtering algorithms that often have higher recall than filtering by phonetic keys or ngrams are *pivot-based retrieval* and *partition filtering*.

Phonetic Key Indexing. In phonetic key indexing, names are indexed by a phonetic representation created by a key function that maps sequences of characters to phonetic categories. Such key functions partition the name space into equivalence classes of names having identical phonetic representations. Each member of a partition is indexed by the shared phonetic representation.

The oldest phonetic key function is apparently Soundex, which was patented in 1918 and 1922 by Russell and Odell (U.S. Patents 1,261,167 and 1,435,663) and described in (Knuth, 1975). Despite Soundex's has many well-known limitations, including inability to handle different first letters with identical pronunciations (*e.g.*, Soundex of "Kris" is K620, but Soundex of "Chris" is C620), truncation of long names, and bias towards English pronunciations, Soundex is still in use in many law enforcement and national security applications (Dizard, 2004). A number of alternative phonetic encodings have been developed in response to the limitations of Soundex, *e.g.*, (Taft, 1970; Gadd, 1990; Zobel and Dart, 1996; Philips, 1990; Philips, 2000; Hodge and Austin, 2001; Christen, 2006). While each of these alternatives has some advantages over Soundex, none is adaptable to alternative distance metrics. For purposes of comparison, Phonex (Gadd, 1990) was included in the evaluation below because it was found to be the most accurate phonetic key for last names in an evaluation by (Christen, 2006).

Ngram Filtering. The second common filtering algorithm for names is ngram indexing, under which each pattern string is indexed by every n-element substring, *i.e.*, every sequence of n contiguous characters occurring in the pattern string (typically, the original string is padded with special leading and trailing characters to distinguish the start and end of the name). The candidates for each target string are retrieved using the ngrams in the target as indices (Cohen et al., 2003). Typical values for n are 3 or 4.

Pivot-Based Retrieval. *Pivot-based* retrieval techniques are applicable to domains, such as name matching, in which entities are not amenable to vector representation but for which the distance metric satisfies the triangle inequality (Chavez et al., 2001).[1]

The key idea is to organize the index around a small group of elements, called *pivots*. In retrieval, the distance between the query probe q and any element e can be estimated based on the distances of each to one or more pivots. There are numerous pivot-based metric space indexing algorithms. An instructive survey of these algorithms is set forth in (Chavez et al., 2001).

One of the oldest, and often best-performing, pivot-based indices is Burkhart-Keller Trees (BKT) (Burkhard and Keller, 1973; Baeza-Yates and Navarro, 1998). BKT is suitable for discrete-valued distance metrics. Construction of a BKT starts with selection of an arbitrary element as the root of the tree. The i^{th} child of the root consists of all elements of distance i from the root. A new BKT is recursively constructed for each child until the number of elements in a child falls below a predefined bucket size.

A range query on a BKT with distance metric d, probe q, range k, and pivot p is performed as follows. If the BKT is a leaf node, then the distance metric d is applied between q and each element of the leaf node, and those elements e for which $d(q, e) < k$ are returned. Otherwise, all subtrees with index i for which $|d(q, e) - i| \leq k$ are recursively searched.

While all names within k of a query are guaranteed to be retrieved by a BKT (*i.e.*, recall is 100%), there are no guarantees on precision. During search, one application of the distance metric is required at each internal node traversed, and a distance-metric application is required for each candidate element in leaf nodes reached during the traversal. The number of nodes searched is exponential in k (Chavez et al., 2001).

Partition Filtering. *Partition filtering* (Wu and Manber, 1991; Navarro and Baeza-Yates, 1999), is an improvement over ngram filtering that relies on the observation that if a pattern string P of length m is divided into segments of length $\lfloor \frac{m}{(k+1)} \rfloor$, then any string that matches P with at most k errors must contain an exact match for at least one of the segments (intuitively, it would take at least k + 1 errors, *e.g.*, edit operations, to alter all of these segments). Strings indexed by $\lfloor \frac{m}{(k+1)} \rfloor$-length segments can be retrieved by an efficient exact string matching algorithm, such as suffix trees or Aho-Corasick trees. This is necessary because partitions, unlike ngrams, vary in length.

Partition filtering differs from ngram filtering in two respects. First, ngrams overlap, whereas partition filtering involves partitioning each string into non-overlapping segments. Second, the choice of n in ngram filtering is typically independent of k, whereas the size of the segments in filtering is chosen based on k. Since in most applications n is independent of k, ngram retrieval, like phonetic key indexing, lacks any guaranteed lower bound on recall, whereas partition filtering guarantees 100% recall when the distance metric is edit distance.

3 The ETK algorithm

3.1 Motivation

Any key function partitions the universe of strings into equivalence classes of strings that share a common key. If a key function is to serve as a filter, matching names must be members of the same equivalence class. However, no single partition can produce equivalence classes that both include all matching pairs and exclude all non-matching pairs.[2]

[1] Edit distance satisfies the triangle inequality because any string A can be transformed into another string C by first transforming A to any other string B, then transforming B into C. Thus, edit-distance(A,C) cannot be greater than edit-distance(A,B) + edit-distance(B,C) for any strings A, B, and C.

[2] For example, suppose that for strings A, B, and C and distance metric d, d(A,B) = .9, d(B,C) = .9, d(A,C) = 1.7, and suppose that 1.0 is the match threshold. A query on A would require a partition that puts A and B in the same equivalence class

A search key that creates partitions in which there is a low probability that non-matching pairs share a common equivalence class will have high precision, although possibly low recall. However, the recall of an ensemble of search keys, each having non-zero recall and each being independent of the others, can be expected to be greater than the recall of any individual key. A high-precision and high-recall index can therefore be constructed if one can find, for a given similarity metric and match threshold, a sufficiently large set of key functions that (1) are independent, (2) each have high-precision under the metric and threshold, and (3) have non-zero recall.

The objective of ETK is to learn a set of independent, high-precision key functions from training data consisting of equivalence classes of names that satisfy the matching criteria. The similarity metric and threshold are implicit in the training data. Thus, under this approach a key function can be learned even if the similarity model is unknown, provided that sufficient equivalence classes are available.

For each equivalence class, ETK attempts to find the shortest transformation rules capable of converting all members of the equivalence class into an identical orthographic representation. The entire collection of transformation rules for all equivalence classes, which in general has many inconsistencies, is then partitioned into separate consistent subsets. Each subset of transformation rules constitutes an independent key function. Each pattern name is indexed by each key produced by applying a key function to it, and the candidate matches for a new name consist of all pattern names that share at least one key.

The equivalence classes of matching names can be obtained either through some *a priori* source (such as alias lists or manual construction) or by applying the similarity metric to pairs in a training set, *e.g.*, repeated leave-one-out retrievals with a known distance metric. In the former case, the keys are purely empirical; in the later the key functions are in effect a way of compiling the distance metric to speed retrieval.

3.2 Procedure

Inducing Transformation Rules. The inductive process starts with a collection of equivalence classes under a given distance metric and match threshold *k*. A collection of transformation rules are derived from these equivalence classes as follows. For each equivalence class EC:

- The element of EC with the least mean pairwise edit distance to the other class members (breaking ties by preferring shorter elements) is selected as the centroid. For example, if EC is {LEIGHTON LAYTON SLEIGHTON}, then LEIGHTON would be the centroid because it has a smaller edit distance to the other elements than they do to each other.

- For each element E other than the centroid, dynamic programming is used to find an alignment of E with the centroid that maximizes the number of corresponding identical characters.[3] For example, the alignment of LEIGHTON and LAYTON would be:

```
LAY--TON
LEIGHTON
```

- For each character c of the centroid, all windows of characters in E of length from 1 to some constant maxWindow centered on the character in the source corresponding to c are found, skipping blank characters. Each mapping from a window to c constitutes a rule. For example, for maxWindow 7 and the alignment above, the transformation rules for the E in LEIGHTON would be:

```
$$LAYTO → E
$LAYT → E
LAY → E
A → E
```

and C into a different equivalence class, a query on C would require a partition that puts B and C in the same equivalence class and A in a different equivalence class, and a query on B would require a partition in which all three were in the same equivalence class. Thus, three independent keys would be needed to satisfy all three queries while excluding non-matching names.

[3] See (Damper et al., 2004) for details on alignment by dynamic programming. The approach taken here assigns a slightly higher association weight for aligned identical consonants than for aligned identical vowels so that, *ceteris paribus*, consonant alignment is preferred to vowel alignment and assigns a slightly higher association weight to non-identical letters that are both vowels or both consonants than to vowel/consonant alignments.

Transformation rules derived from multiple equivalence classes typically have many inconsistencies, *i.e.*, rules with identical left-hand sides (LHSs) but different right hand sides (RHSs). All RHSs for a given LHS are grouped together and ranked by frequency of occurrence in the training data. For example, the frequency of alternative rules for the middle characters LAN and LEI for the U.S. name pronunciation set with $k = 1$ discussed below is:

LAY → E	5	LEI → A	2
LAY → A	4	LEI → E	1
LAY → AN	3	LEI → -	1

Key Formation. The transformation rules are subdivided two different ways: by LHS, *e.g.*, separating rules for LAY from those for LEI, and by RHS frequency, *e.g.*, separating LAY → E (the most frequent rule for LAY) from LAY→ A (the next most frequent). The highest frequency RHS rules from the example above are:

LAY → E
LEI → A

and the next most frequent are:

LAY → A
LEI → E

If rules are divided into l LHS subsets, and each subset is further subdivided by taking the r highest ranked RHSs (with RHSs ranked lower than r ignored), the result is a total of lr subsets. Each of these lr subsets defines a key function. For each position in a word to which the key function is to be applied (padded with leading and training markers), the rule with the longest (*i.e.*, most specific) LHS that matches the window centered at that position is used to determine the corresponding character in the key. If no rules apply, the character in the key is the same as that in the original word.

For example, suppose that the word to which the key is to be applied is CREIGHTON and transformations include LEIGHTO → -, EIGHT → - and IGH → G. The character in the key corresponding to the G in CREIGHTON would be - (*i.e.*, a deletion) because the EIGHT is the longest LHS matching at that position. The key consists of the concatenation of the RHSs produced by successively applying the key function to each position in the orginal word.

This procedure is similar to window-based pronunciation-learning algorithms, *e.g.*, (Sejnowski and Rosenberg, 1987; Bakiri and Dietterich, 1999), but differs in that the objective is not determining a correct pronunciation, but is instead transforming words that are similar under a given metric into a single, consistent orthographic representation.

3.3 Filtering with ETK

The lr subsets of transformation rules induced from a given set of equivalence classes define an ensemble of key functions. To filter potential matches with this ensemble, each pattern is added to a hash table indexed by each key generated by a key function. Candidate matches to a text string consist of all patterns indexed by the keys generated from the text by the ensemble of key functions. For example, suppose that (as is the case with the rule sets for American names, pronunciation distance, and $k = 0$) patterns ROLLINS and ROWLAND have keys that include {ROWLINS ROLINS} and {RONLLAND ROLAN}, respectively, and that text RAWLINS has keys that include {ROWLINS RALINS}. Then ROLLINS but not ROWLAND would be retrieved because it is indexed by a key shared with ROWLINS.[4]

4 Evaluation

The retrieval accuracy of ETK was compared to that of BKT, filtering by partition, ngram filtering, Phonex, and Soundex on sets of U.S. and German names. The U.S. name set consisted of the 5,000 most common last names identified during the most recent U.S. Census[5] which have pronunciations in cmudict, the CMU pronouncing dictionary.[6] The German name set consisted of the first 5963 entries in the HADI-BOMP collection[7] whose part of speech is NAM.

The filtering algorithms were compared with respect to two alternative distance metrics. The first was *pronunciation distance*, which consists of edit

[4] In the evaluation below, the original string itself is added as an additional index key. This addition slightly increases both recall and precision.

[5] The names were taken from the 1990 U.S. Census collection of 88,799 last names at http://www.census.gov/genealogy/names/names_files.html.

[6] http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[7] http://www.ikp.uni-bonn.de/dt/forsch/phonetik/bomp.

distance between pronunciations represented using the cmudict phoneme set for U.S. names and the HADI-BOMP phoneme set for German names. Stress values were removed from cmudict pronunciations, and syllable divisions were removed from HADI-BOMP pronunciations. When there were multiple pronunciations for a name in cmudict, the first was used. In cmudict, for example, MEUSE and MEWES have pronunciation distance of 0 because both have pronunciation M Y UW Z. In HADI-BOMP, HELGARD and HERBART have pronunciation distance 2 because their pronunciations are h E l g a r t and h E r b a r t. The second distance metric was edit distance with unit weights for insertions, deletions, and substitutions. In practice, appropriate distance metrics might be Jaro (Jaro, 1995), Winkler (Winkler, 1999), or some metric specialized to a particular phonetic or error model. Pronunciation and edit distance were chosen as representative of phonetic and non-phonetic metrics.

Training data for ETK for a given language, match threshold k, and similarity metric consisted of all sets of at least 2 names containing only elements were within k of some element of the set under the metric. These training sets were created by performing a retrieval on every name in each collection using BKT, which has 100% recall. For each retrieval, the true positives from BKT's return set were determined by applying the similarity metric between each return set element and the query. If there were at least 2 true positives (including the query itself), the set of true positives was included in the training set.[8]

ETK was tested using cross validation, so that names in the training set and those in the testing set were disjoint. Specifically, all names in the testing set were removed from each collection in the training set. If at least 2 names remained, the collection was retained. ETK's maxWindow size was 7, as in the examples above.

In BKT, the bucket size (maximum number of elements in any leaf node) was 2, and the longest element (rather than a random element) was selected as the root of each subtree. The rationale for this choice is that there is typically more variance in distance from a longer word than from a shorter word, and greater variance increases the branching factor in BKT, reducing tree depth and therefore the number of nodes visited during search.

Since the importance of precision in filtering is that it determines the number of calls to the similarity metric required for a given level of recall, precision figures for BKT include *internal* calls to the similarity metric, that is, calls during indexing. Thus, precision of BKT is the number of true positives divided by the number of all positives plus the number of internal metric calls.

In Soundex and Phonex indexing, each name was indexed by its Soundex (Phonex) key. Similarly, in ngram filtering each name was indexed by all its ngrams, with special leading and trailing characters added. Retrieval was performed by finding the Soundex or Phonex encoding or the ngrams of each query and retrieving every name indexed by the Soundex or Phonex encoding or any ngram. Precision was measured with duplicates removed.

In partition filtering, each name was indexed by each of its $k + 1$ partitions, and the partitions themselves were organized in an Aho-Curasick tree (Gusfield, 1999). Retrieval was performed by applying the Aho-Curasick tree to the query to determine all partitions occurring in the query and retrieving the names corresponding to each partition, removing duplicates.

4.1 Optimizing LHS and RHS Subdivisions

The first experiment was performed to clarify the optimal sizes of l, the number of LHS subdivisions, and r, the number of RHS ranks. ETK was tested on the U.S. name set with $k = 1$, pronunciation distance as similarity metric, and 10-fold cross validation for $l \in \{1, 2, 4, 8, 16, 32\}$ and $r \in \{1, 2\}$.

As shown in Table 1, when $l = 1$, $r = 2$ has higher f-measure than $r = 1$, but when l is 2 or greater, the best value for r is 1. Overall, the highest f-measure is obtained with $l = 8$ and $r = 1$. In the experiments below, the value of 16 was used for l because this leads to slightly higher recall at a small cost in decreased f-measure.

[8] Note that each set of true positives is a cluster having the query as its centroid and radius k under the distance metric. The triangle inequality guarantees that the maximum distance between any pair of names in the collection is no greater than $2k$.

Table 1: F-measure for $l \in \{1, 2, 4, 8, 16, 32\}$ and $r \in \{1, 2, \}$ on U.S. names with pronunciation distance and $k = 1$ in 10-fold cross validation.

	1	2	4	8	16	32
1	0.1431	**0.2112**	**0.3039**	**0.3550**	**0.3428**	**0.2928**
2	**0.1469**	0.1858	0.1520	0.0729	0.0264	0.0108

4.2 Comparison of ETK to Other Filter Algorithms

The retrieval accuracy of ETK was compared to that of BKT, partition, ngram, Phonex, and Soundex on the U.S. and German name sets for pronunciation distance with $k \in \{0, 1, 2\}$ and for edit distance with $k \in \{1, 2\}$. In tests involving pronunciation distance BKT was tested under two conditions: with the pronunciation distance function available to BKT during indexing and retrieval; and the distance function unavailable, so that BKT indexing and retrieval was performed on the surface form even though the actual similarity metric was pronunciation distance. This is intended to simulate the situation in which examples of matching names are available but the underlying similarity metric is unknown. Ngram and partition filtering were performed on letters only.

Tables 2 and 3 show recall, precision, and f-measure for pronunciation distance on U.S. and German names, respectively, with $k \in \{0, 1, 2\}$, $l = 16$, and $r = 1$. ETK has the highest f-measure under all conditions because its precision is consistently higher than that of the other algorithms. This is because each key function in ETK applies only transformations representing orthographic differences between names in the same equivalence class. Thus, the transformations are very conservative. BKT always has recall of 1.0 when the pronunciation model is available, but in many cases a model may be unavailable. When no model is available, no single algorithm consistently has the highest recall. Ngrams, partition, Phonex, and BKT each had the highest recall in at least one language/error threshold combination.

Table 2: Recall, precision, and f-measure for pronunciation distance on U.S. surnames. K is maximum permitted error. BKT-NM is BKT without the pronunciation model. Best results are shown in bold, including highest recall in addition to BKT.

		recall	precision	f-measure
k=0	BKT	**1.0000**	0.0152	0.0299
	BKT-NM	0.0510	0.0003	0.0006
	partition	0.1298	0.0168	0.0298
	soundex	0.8350	0.0331	0.0637
	phonex	**0.8811**	0.0173	0.0339
	ngrams	0.7457	0.0034	0.0068
	ETK	0.5642	**0.3314**	**0.4175**
k=1	BKT	**1.0000**	0.0039	0.0078
	BKT-NM	0.5704	0.0019	0.0038
	partition	**0.6157**	0.0092	0.0181
	soundex	0.4422	0.1803	0.2562
	phonex	0.4969	0.1008	0.1676
	ngrams	0.4453	0.0213	0.0406
	ETK	0.4862	**0.2647**	**0.3428**
k=2	BKT	**1.0000**	0.0088	0.0174
	BKT-NM	**0.7588**	0.0050	0.0099
	partition	0.6948	0.0122	0.0240
	soundex	0.1298	**0.4350**	0.2000
	phonex	0.1708	0.2860	0.2139
	ngrams	0.2063	0.0825	0.1178
	ETK	0.4502	0.1953	**0.2724**

Tables 4 and 5 show recall, precision, and f-measure for edit distance on U.S. and German names, respectively, with $k \in \{1, 2\}$, $l = 16$, and $r = 1$ ($k = 0$ would be an exact match on the surface form, for which all algorithms would have recall 1.0). Again, ETK has the highest f-measure because of its consistently high precision.

4.3 Training Set Size

The sensitivity of ETK to training set size was tested by performing 50-fold cross-validation with training sets for pronunciation distance on U.S. names of sizes in {48, 96, 191, 381, 762, 1524, 3047} drawn from the 3047 equivalence classes in the 5000 U.S. names with pronunciation distance and $k = 1$. As shown in Figure 1, the learning curve rises steeply for the entire range of training set sizes considered in this experiment.

5 Conclusion

The experimental results demonstrate the feasibility of basing search keys on transformation rules acquired from examples. If sufficient examples of names that match under a given distance metric and error threshold are available, keys can be induced that lead to good performance in comparison to alternative filtering algorithms. Moreover, the results involving pronunciation distance illustrate how phonetic keys can be learned that are specific to indi-

Table 3: Recall, precision, and f-measure for pronunciation distance on German names. K is maximum permitted error. Best results are shown in bold.

		recall	precision	f-measure
k=0	BKT	**1.0000**	0.0056	0.0110
	BKT-NM	0.1600	0.0003	0.0007
	partition	0.1223	0.0059	0.0112
	soundex	0.7059	0.0125	0.0235
	phonex	0.8997	0.0061	0.0122
	ngrams	**0.9348**	0.0016	0.0031
	ETK	0.7715	**0.3606**	**0.4915**
k=1	BKT	**1.0000**	0.0013	0.0026
	BKT-NM	**0.7923**	0.0006	0.0013
	partition	0.7865	0.0031	0.0062
	soundex	0.3969	0.0533	0.0940
	phonex	0.5048	0.0270	0.0512
	ngrams	0.6866	0.0090	0.0178
	ETK	0.5503	**0.3820**	**0.4510**
k=2	BKT	**1.0000**	0.0018	0.0037
	BKT-NM	**0.8533**	0.0010	0.0021
	partition	0.8384	0.0029	0.0058
	soundex	0.1311	0.1209	0.1258
	phonex	0.1693	0.0640	0.0929
	ngrams	0.2801	0.0255	0.0468
	ETK	0.3496	**0.1687**	**0.2276**

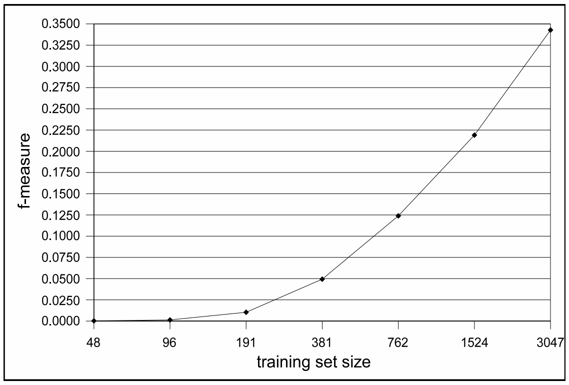
Table 4: Recall, precision, and f-measure for edit distance on U.S. surnames.

		recall	precision	f-measure
k=0	BKT	**1.0000**	0.0024	0.0048
	partition	**1.0000**	0.0106	0.0210
	soundex	0.3537	0.1010	0.1572
	phonex	0.3937	0.0564	0.0986
	ngrams	0.8408	0.0288	0.0557
	ETK	0.6768	**0.3244**	**0.4386**
k=1	BKT	**1.0000**	0.0052	0.0103
	partition	**1.0000**	0.0139	0.0275
	soundex	0.1038	0.2692	0.1498
	phonex	0.1288	0.1696	0.1464
	ngrams	0.4112	0.1300	0.1976
	ETK	0.4001	**0.3565**	**0.3770**

Table 5: Recall, precision, and f-measure for edit distance on German names.

		recall	precision	f-measure
k=0	BKT	**1.0000**	0.0009	0.0018
	partition	**1.0000**	0.0045	0.0091
	soundex	0.5266	0.0826	0.1429
	phonex	0.6101	0.0374	0.0704
	ngrams	0.8880	0.0134	0.0264
	ETK	0.6647	**0.4957**	**0.5679**
k=1	BKT	**1.0000**	0.0017	0.0034
	partition	**1.0000**	0.0048	0.0096
	soundex	0.1592	0.2052	0.1793
	phonex	0.2019	0.1063	0.1392
	ngrams	0.4036	0.0516	0.0915
	ETK	0.3986	**0.3466**	**0.3708**

Figure 1: F-measure for U.S. names for training sets containing varying numbers of collections, with $k = 1$, $l = 16$, and $r = 1$. Each training instance consists of all names within k of some centriod under the metric.



vidual match criteria. In filtering under pronunciation distance, ETK's f-measure for German names was similar to its f-measure for U.S. names (actually higher for $k \in \{0, 1\}$) whereas Soundex and Phonex were approximately an order of magnitude lower.

Although ETK consistently had the highest f-measure in this experiment, it does not follow that ETK is necessarily the most desirable name filter for any particular application. In many applications recall may be much more important than precision. In such cases, it may be essential to choose the highest recall algorithm notwithstanding a lower f-measure. However, the highest recall algorthms can lead to a very large number of distance-metric applications. For example, in some data sets the number of nodes examined by BKT during retrieval is a significant proportion of the entire pattern set.

ETK has the disadvantage of requiring a large set of training examples consisting of equivalence sets of strings that match under the metric and maximum allowable error. Where such large numbers of equivalence sets are unavailable, it may be better to use simpler and less-informed filters.

A number of variations of ETK are possible. For example, keys could consist of finite-state transducers trained from consistent subsets of mappings rather than transformation rules. There are also many possible alternatives to ETK's window-based approach to deriving mappings from examples.

In summary, this work has demonstrated that ensembles of keys induced from equivalence classes of names under a specific distance metric and maximum allowable error can filter names with high f-measure. The experimental results illustrate the benefits both of acquiring keys that are adapted to specific similarity criteria and of indexing with multiple independent keys.

References

R. A. Baeza-Yates and G. Navarro. 1998. Fast approximate string matching in a dictionary. In *String Processing and Information Retrieval*, pages 14–22.

G. Bakiri and T. Dietterich. 1999. Achieving high-accuracy text-to-speech with machine learning. *Data mining in speech synthesis*.

M. Bilenko, W. W. Cohen, S. Fienberg, R. J. Mooney, and P. Ravikumar. 2003. Adaptive name-matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

W. A. Burkhard and R. M. Keller. 1973. Some approaches to best-match file searching. *Commun. ACM*, 16(4):230–236.

E. Chavez, G. Navarro, R. A. Baeza-Yates, and J. L. Marroquin. 2001. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321.

P. Christen. 2006. A comparison of personal name matching: Techniques and practical issues. In *Proceedings of the ICDM 2006 Workshop on Mining Complex Data (MCD)*, December.

W. W. Cohen, P. Ravikumar, and S. E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 73–78, Acapulco, Mexico, August.

R. I. Damper, Y. Marchand, J. D. S. Marsters, and A. I. Bazin. 2004. Aligning letters and phonemes for speech synthesis. In *Proceedings of 5th International Speech Communication Association (ISCA) Workshop on Speech Synthesis*, pages 209–214, Pittsburgh, PA.

W. Dizard. 2004. Obsolete algorithm tangles terrorst/criminal watch lists. *Government Computer News*, 23(12), August 17.

T. Gadd. 1990. Phonix: The algorithm. *Program: automated library and information systems*, 24(4).

D. Gentner and K. Forbus. 1991. MAC/FAC: A model of similarity-based retrieval. In *Thirteenth Annual Conference of the Cognitive Science Society*, pages 504–509.

D. Gusfield. 1999. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press.

V. J. Hodge and J. Austin. 2001. An evaluation of phonetic spell checkers. Technical report, Department of Computer Science, University of York.

M. A. Jaro. 1995. Probabilistic linkage of large public health data files. *Statistics in Medicine*, 14(5–7):491–498.

D. E. Knuth. 1975. *Fundamental Algorithms*, volume III of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts.

G. Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 288–295, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

G. Navarro and R. Baeza-Yates. 1999. Very fast and simple approximate string matching. *Information Processing Letters*, 72:65–70.

G. Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, March.

L. Philips. 1990. Hanging on the metaphone. *Computer Language Magazine*, 7(12), December.

L. Philips. 2000. The double metaphone search algorithm. *C/C++ Users Journal*, 18(1), June 1.

E. S. Ristad and P. N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

T.J Sejnowski and C.R. Rosenberg. 1987. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168.

D. Shin and B. Zhang. 1998. A two-stage retrieval model for the TREC-7 ad hoc task. In *Text REtrieval Conference*, pages 439–445.

R. Taft. 1970. Name search techniques: New york state identification and intelligence system. Technical Report 1, State of New York.

W. E. Winkler. 1999. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC.

S. Wu and U. Manber. 1991. Fast text searching with errors. Technical Report TR-91-11, University of Arizona.

J. Zobel and P. Dart. 1996. Phonetic string matching: lessons from information retrieval. *SIGIR Forum*, 166–172.

The CoNLL 2007 Shared Task on Dependency Parsing

Joakim Nivre[*†] **Johan Hall**[*] **Sandra Kübler**[‡] **Ryan McDonald**[**]
Jens Nilsson[*] **Sebastian Riedel**[††] **Deniz Yuret**[‡‡]

[*]Växjö University, School of Mathematics and Systems Engineering, *first.last*@vxu.se
[†]Uppsala University, Dept. of Linguistics and Philology, joakim.nivre@lingfil.uu.se
[‡]Indiana University, Department of Linguistics, skuebler@indiana.edu
[**]Google Inc., ryanmcd@google.com
[††]University of Edinburgh, School of Informatics, S.R.Riedel@sms.ed.ac.uk
[‡‡]Koç University, Dept. of Computer Engineering, dyuret@ku.edu.tr

Abstract

The Conference on Computational Natural Language Learning features a shared task, in which participants train and test their learning systems on the same data sets. In 2007, as in 2006, the shared task has been devoted to dependency parsing, this year with both a multilingual track and a domain adaptation track. In this paper, we define the tasks of the different tracks and describe how the data sets were created from existing treebanks for ten languages. In addition, we characterize the different approaches of the participating systems, report the test results, and provide a first analysis of these results.

1 Introduction

Previous shared tasks of the Conference on Computational Natural Language Learning (CoNLL) have been devoted to chunking (1999, 2000), clause identification (2001), named entity recognition (2002, 2003), and semantic role labeling (2004, 2005). In 2006 the shared task was multilingual dependency parsing, where participants had to train a single parser on data from thirteen different languages, which enabled a comparison not only of parsing and learning methods, but also of the performance that can be achieved for different languages (Buchholz and Marsi, 2006).

In dependency-based syntactic parsing, the task is to derive a syntactic structure for an input sentence by identifying the syntactic *head* of each word in the sentence. This defines a *dependency graph*, where the nodes are the words of the input sentence and the arcs are the binary relations from head to dependent. Often, but not always, it is assumed that all words except one have a syntactic head, which means that the graph will be a tree with the single independent word as the root. In *labeled* dependency parsing, we additionally require the parser to assign a specific type (or label) to each dependency relation holding between a head word and a dependent word.

In this year's shared task, we continue to explore data-driven methods for multilingual dependency parsing, but we add a new dimension by also introducing the problem of domain adaptation. The way this was done was by having two separate tracks: a multilingual track using essentially the same setup as last year, but with partly different languages, and a domain adaptation track, where the task was to use machine learning to adapt a parser for a single language to a new domain. In total, test results were submitted for twenty-three systems in the multilingual track, and ten systems in the domain adaptation track (six of which also participated in the multilingual track). Not everyone submitted papers describing their system, and some papers describe more than one system (or the same system in both tracks), which explains why there are only (!) twenty-one papers in the proceedings.

In this paper, we provide task definitions for the two tracks (section 2), describe data sets extracted from available treebanks (section 3), report results for all systems in both tracks (section 4), give an overview of approaches used (section 5), provide a first analysis of the results (section 6), and conclude with some future directions (section 7).

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 915–932,
Prague, June 2007. ©2007 Association for Computational Linguistics

2 Task Definition

In this section, we provide the task definitions that were used in the two tracks of the CoNLL 2007 Shard Task, the multilingual track and the domain adaptation track, together with some background and motivation for the design choices made. First of all, we give a brief description of the data format and evaluation metrics, which were common to the two tracks.

2.1 Data Format and Evaluation Metrics

The data sets derived from the original treebanks (section 3) were in the same column-based format as for the 2006 shared task (Buchholz and Marsi, 2006). In this format, sentences are separated by a blank line; a sentence consists of one or more tokens, each one starting on a new line; and a token consists of the following ten fields, separated by a single tab character:

1. ID: Token counter, starting at 1 for each new sentence.

2. FORM: Word form or punctuation symbol.

3. LEMMA: Lemma or stem of word form, or an underscore if not available.

4. CPOSTAG: Coarse-grained part-of-speech tag, where the tagset depends on the language.

5. POSTAG: Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.

6. FEATS: Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.

7. HEAD: Head of the current token, which is either a value of ID or zero (0). Note that, depending on the original treebank annotation, there may be multiple tokens with HEAD=0.

8. DEPREL: Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that, depending on the original treebank annotation, the dependency relation when HEAD=0 may be meaningful or simply ROOT.

9. PHEAD: Projective head of current token, which is either a value of ID or zero (0), or an underscore if not available.

10. PDEPREL: Dependency relation to the PHEAD, or an underscore if not available.

The PHEAD and PDEPREL were not used at all in this year's data sets (i.e., they always contained underscores) but were maintained for compatibility with last year's data sets. This means that, in practice, the first six columns can be considered as *input* to the parser, while the HEAD and DEPREL fields are the *output* to be produced by the parser. Labeled training sets contained all ten columns; blind test sets only contained the first six columns; and gold standard test sets (released only after the end of the test period) again contained all ten columns. All data files were encoded in UTF-8.

The official evaluation metric in both tracks was the *labeled attachment score* (LAS), i.e., the percentage of tokens for which a system has predicted the correct HEAD and DEPREL, but results were also reported for *unlabeled attachment score* (UAS), i.e., the percentage of tokens with correct HEAD, and the *label accuracy* (LA), i.e., the percentage of tokens with correct DEPREL. One important difference compared to the 2006 shared task is that all tokens were counted as "scoring tokens", including in particular all punctuation tokens. The official evaluation script, eval07.pl, is available from the shared task website.[1]

2.2 Multilingual Track

The multilingual track of the shared task was organized in the same way as the 2006 task, with annotated training and test data from a wide range of languages to be processed with one and the same parsing system. This system must therefore be able to learn from training data, to generalize to unseen test data, and to handle multiple languages, possibly by adjusting a number of hyper-parameters. Participants in the multilingual track were expected to submit parsing results for all languages involved.

[1] http://depparse.uvt.nl/depparse-wiki/SoftwarePage

One of the claimed advantages of dependency parsing, as opposed to parsing based on constituent analysis, is that it extends naturally to languages with free or flexible word order. This explains the interest in recent years for multilingual evaluation of dependency parsers. Even before the 2006 shared task, the parsers of Collins (1997) and Charniak (2000), originally developed for English, had been adapted for dependency parsing of Czech, and the parsing methodology proposed by Kudo and Matsumoto (2002) and Yamada and Matsumoto (2003) had been evaluated on both Japanese and English. The parser of McDonald and Pereira (2006) had been applied to English, Czech and Danish, and the parser of Nivre et al. (2007) to ten different languages. But by far the largest evaluation of multilingual dependency parsing systems so far was the 2006 shared task, where nineteen systems were evaluated on data from thirteen languages (Buchholz and Marsi, 2006).

One of the conclusions from the 2006 shared task was that parsing accuracy differed greatly between languages and that a deeper analysis of the factors involved in this variation was an important problem for future research. In order to provide an extended empirical foundation for such research, we tried to select the languages and data sets for this year's task based on the following desiderata:

- The selection of languages should be typologically varied and include both new languages and old languages (compared to 2006).
- The creation of the data sets should involve as little conversion as possible from the original treebank annotation, meaning that preference should be given to treebanks with dependency annotation.
- The training data sets should include at least 50,000 tokens and at most 500,000 tokens.[2]

The final selection included data from Arabic, Basque, Catalan, Chinese, Czech, English, Greek, Hungarian, Italian, and Turkish. The treebanks from which the data sets were extracted are described in section 3.

2.3 Domain Adaptation Track

One well known characteristic of data-driven parsing systems is that they typically perform much worse on data that does not come from the training domain (Gildea, 2001). Due to the large overhead in annotating text with deep syntactic parse trees, the need to adapt parsers from domains with plentiful resources (e.g., news) to domains with little resources is an important problem. This problem is commonly referred to as *domain adaptation*, where the goal is to adapt annotated resources from a *source domain* to a *target domain* of interest.

Almost all prior work on domain adaptation assumes one of two scenarios. In the first scenario, there are limited annotated resources available in the target domain, and many studies have shown that this may lead to substantial improvements. This includes the work of Roark and Bacchiani (2003), Florian et al. (2004), Chelba and Acero (2004), Daumé and Marcu (2006), and Titov and Henderson (2006). Of these, Roark and Bacchiani (2003) and Titov and Henderson (2006) deal specifically with syntactic parsing. The second scenario assumes that there are no annotated resources in the target domain. This is a more realistic situation and is considerably more difficult. Recent work by McClosky et al. (2006) and Blitzer et al. (2006) have shown that the existence of a large unlabeled corpus in the new domain can be leveraged in adaptation. For this shared-task, we are assuming the latter setting – *no annotated resources in the target domain*.

Obtaining adequate annotated syntactic resources for multiple languages is already a challenging problem, which is only exacerbated when these resources must be drawn from multiple and diverse domains. As a result, the only language that could be feasibly tested in the domain adaptation track was English.

The setup for the domain adaptation track was as follows. Participants were provided with a large annotated corpus from the source domain, in this case sentences from the Wall Street Journal. Participants were also provided with data from three different target domains: biomedical abstracts (development data), chemical abstracts (test data 1), and parent-child dialogues (test data 2). Additionally, a large

[2] The reason for having an upper bound on the training set size was the fact that, in 2006, some participants could not train on all the data for some languages because of time limitations. Similar considerations also led to the decision to have a smaller number of languages this year (ten, as opposed to thirteen).

unlabeled corpus for each data set (training, development, test) was provided. The goal of the task was to use the annotated source data, plus any unlabeled data, to produce a parser that is accurate for each of the test sets from the target domains.[3]

Participants could submit systems in either the "open" or "closed" class (or both). The closed class requires a system to use only those resources provided as part of the shared task. The open class allows a system to use additional resources provided those resources are not drawn from the same domain as the development or test sets. An example might be a part-of-speech tagger trained on the entire Penn Treebank and not just the subset provided as training data, or a parser that has been hand-crafted or trained on a different training set.

3 Treebanks

In this section, we describe the treebanks used in the shared task and give relevant information about the data sets created from them.

3.1 Multilingual Track

Arabic The analytical syntactic annotation of the Prague Arabic Dependency Treebank (PADT) (Hajič et al., 2004) can be considered a pure dependency annotation. The conversion, done by Otakar Smrz, from the original format to the column-based format described in section 2.1 was therefore relatively straightforward, although not all the information in the original annotation could be transfered to the new format. PADT was one of the treebanks used in the 2006 shared task but then only contained about 54,000 tokens. Since then, the size of the treebank has more than doubled, with around 112,000 tokens. In addition, the morphological annotation has been made more informative. It is also worth noting that the parsing units in this treebank are in many cases larger than conventional sentences, which partly explains the high average number of tokens per "sentence" (Buchholz and Marsi, 2006).

Basque For Basque, we used the 3LB Basque treebank (Aduriz et al., 2003). At present, the treebank consists of approximately 3,700 sentences, 334 of which were used as test data. The treebank comprises literary and newspaper texts. It is annotated in a dependency format and was converted to the CoNLL format by a team led by Koldo Gojenola.

Catalan The Catalan section of the CESS-ECE Syntactically and Semantically Annotated Corpora (Martí et al., 2007) is annotated with, among other things, constituent structure and grammatical functions. A head percolation table was used for automatically converting the constituent trees into dependency trees. The original data only contains functions related to the verb, and a function table was used for deriving the remaining syntactic functions. The conversion was performed by a team led by Lluís Màrquez and Antònia Martí.

Chinese The Chinese data are taken from the Sinica treebank (Chen et al., 2003), which contains both syntactic functions and semantic functions. The syntactic head was used in the conversion to the CoNLL format, carried out by Yu-Ming Hsieh and the organizers of the 2006 shared task, and the syntactic functions were used wherever it was possible. The training data used is basically the same as for the 2006 shared task, except for a few corrections, but the test data is new for this year's shared task. It is worth noting that the parsing units in this treebank are sometimes smaller than conventional sentence units, which partly explains the low average number of tokens per "sentence" (Buchholz and Marsi, 2006).

Czech The analytical syntactic annotation of the Prague Dependency Treebank (PDT) (Böhmová et al., 2003) is a pure dependency annotation, just as for PADT. It was also used in the shared task 2006, but there are two important changes compared to last year. First, version 2.0 of PDT was used instead of version 1.0, and a conversion script was created by Zdenek Zabokrtsky, using the new XML-based format of PDT 2.0. Secondly, due to the upper bound on training set size, only sections 1–3 of PDT constitute the training data, which amounts to some 450,000 tokens. The test data is a small subset of the development test set of PDT.

[3] Note that annotated development data for the target domain was only provided for the development domain, biomedical abstracts. For the two test domains, chemical abstracts and parent-child dialogues, the only annotated data sets were the gold standard test sets, released only after test runs had been submitted.

English For English we used the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993). In particular, we used sections 2-11 for training and a subset of section 23 for testing. As a preprocessing stage we removed many functions tags from the non-terminals in the phrase structure representation to make the representations more uniform with out-of-domain test sets for the domain adaptation track (see section 3.2). The resulting data set was then converted to dependency structures using the procedure described in Johansson and Nugues (2007a). This work was done by Ryan McDonald.

Greek The Greek Dependency Treebank (GDT) (Prokopidis et al., 2005) adopts a dependency structure annotation very similar to those of PDT and PADT, which means that the conversion by Prokopis Prokopidis was relatively straightforward. GDT is one of the smallest treebanks in this year's shared task (about 65,000 tokens) and contains sentences of Modern Greek. Just like PDT and PADT, the treebank contains more than one level of annotation, but we only used the analytical level of GDT.

Hungarian For the Hungarian data, the Szeged treebank (Csendes et al., 2005) was used. The treebank is based on texts from six different genres, ranging from legal newspaper texts to fiction. The original annotation scheme is constituent-based, following generative principles. It was converted into dependencies by Zóltan Alexin based on heuristics.

Italian The data set used for Italian is a subset of the balanced section of the Italian Syntactic-Semantic Treebank (ISST) (Montemagni et al., 2003) and consists of texts from the newspaper *Corriere della Sera* and from periodicals. A team led by Giuseppe Attardi, Simonetta Montemagni, and Maria Simi converted the annotation to the CoNLL format, using information from two different annotation levels, the constituent structure level and the dependency structure level.

Turkish For Turkish we used the METU-Sabancı Turkish Treebank (Oflazer et al., 2003), which was also used in the 2006 shared task. A new test set of about 9,000 tokens was provided by Gülşen Eryiğit (Eryiğit, 2007), who also handled the conversion to the CoNLL format, which means that we could use all the approximately 65,000 tokens of the original treebank for training. The rich morphology of Turkish requires the basic tokens in parsing to be inflectional groups (IGs) rather than words. IGs of a single word are connected to each other deterministically using dependency links labeled DERIV, referred to as word-internal dependencies in the following, and the FORM and the LEMMA fields may be empty (they contain underscore characters in the data files). Sentences do not necessarily have a unique root; most internal punctuation and a few foreign words also have HEAD=0.

3.2 Domain Adaptation Track

As mentioned previously, the source data is drawn from a corpus of news, specifically the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993). This data set is identical to the English training set from the multilingual track (see section 3.1).

For the target domains we used three different labeled data sets. The first two were annotated as part of the PennBioIE project (Kulick et al., 2004) and consist of sentences drawn from either biomedical or chemical research abstracts. Like the source WSJ corpus, this data is annotated using the Penn Treebank phrase structure scheme. To convert these sets to dependency structures we used the same procedure as before (Johansson and Nugues, 2007a). Additional care was taken to remove sentences that contained non-WSJ part-of-speech tags or non-terminals (e.g., HYPH part-of-speech tag indicating a hyphen). Furthermore, the annotation scheme for gaps and traces was made consistent with the Penn Treebank wherever possible. As already mentioned, the biomedical data set was distributed as a development set for the training phase, while the chemical data set was only used for final testing.

The third target data set was taken from the CHILDES database (MacWhinney, 2000), in particular the EVE corpus (Brown, 1973), which has been annotated with dependency structures. Unfortunately the dependency labels of the CHILDES data were inconsistent with those of the WSJ, biomedical and chemical data sets, and we therefore opted to only evaluate unlabeled accuracy for this data set. Furthermore, there was an inconsistency in how main and auxiliary verbs were annotated for this data set relative to others. As a result of this, submitting

Multilingual											Domain adaptation	
	Ar	Ba	Ca	Ch	Cz	En	Gr	Hu	It	Tu	PCHEM	CHILDES
Language family	Sem.	Isol.	Rom.	Sin.	Sla.	Ger.	Hel.	F.-U.	Rom.	Tur.	Ger.	
Annotation	d	d	c+f	c+f	d	c+f	d	c+f	c+f	d	c+f	d
Training data											Development data	
Tokens (k)	112	51	431	337	432	447	65	132	71	65	5	
Sentences (k)	2.9	3.2	15.0	57.0	25.4	18.6	2.7	6.0	3.1	5.6	0.2	
Tokens/sentence	38.3	15.8	28.8	5.9	17.0	24.0	24.2	21.8	22.9	11.6	25.1	
LEMMA	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	No	
No. CPOSTAG	15	25	17	13	12	31	18	16	14	14	25	
No. POSTAG	21	64	54	294	59	45	38	43	28	31	37	
No. FEATS	21	359	33	0	71	0	31	50	21	78	0	
No. DEPREL	29	35	42	69	46	20	46	49	22	25	18	
No. DEPREL H=0	18	17	1	1	8	1	22	1	1	1	1	
% HEAD=0	8.7	9.7	3.5	16.9	11.6	4.2	8.3	4.6	5.4	12.8	4.0	
% HEAD left	79.2	44.5	60.0	24.7	46.9	49.0	44.8	27.4	65.0	3.8	50.0	
% HEAD right	12.1	45.8	36.5	58.4	41.5	46.9	46.9	68.0	29.6	83.4	46.0	
HEAD=0/sentence	3.3	1.5	1.0	1.0	2.0	1.0	2.0	1.0	1.2	1.5	1.0	
% Non-proj. arcs	0.4	2.9	0.1	0.0	1.9	0.3	1.1	2.9	0.5	5.5	0.4	
% Non-proj. sent.	10.1	26.2	2.9	0.0	23.2	6.7	20.3	26.4	7.4	33.3	8.0	
Punc. attached	S	S	A	S	S	A	S	A	A	S	A	
DEPRELS for punc.	10	13	6	29	16	13	15	1	10	12	8	
Test data											PCHEM	CHILDES
Tokens	5124	5390	5016	5161	4724	5003	4804	7344	5096	4513	5001	4999
Sentences	131	334	167	690	286	214	197	390	249	300	195	666
Tokens/sentence	39.1	16.1	30.0	7.5	16.5	23.4	24.4	18.8	20.5	15.0	25.6	12.9
% New words	12.44	24.98	4.35	9.70	12.58	3.13	12.43	26.10	15.07	36.29	31.33	6.10
% New lemmas	2.82	11.13	3.36	n/a	5.28	n/a	5.82	14.80	8.24	9.95	n/a	n/a

Table 1: Characteristics of the data sets for the 10 languages of the multilingual track and the development set and the two test sets of the domain adaptation track.

results for the CHILDES data was considered optional. Like the chemical data set, this data set was only used for final testing.

Finally, a large corpus of unlabeled in-domain data was provided for each data set and made available for training. This data was drawn from the WSJ, PubMed.com (specific to biomedical and chemical research literature), and the CHILDES data base. The data was tokenized to be as consistent as possible with the WSJ training set.

3.3 Overview

Table 1 describes the characteristics of the data sets. For the multilingual track, we provide statistics over the training and test sets; for the domain adaptation track, the statistics were extracted from the development set. Following last year's shared task practice (Buchholz and Marsi, 2006), we use the following definition of projectivity: An arc (i, j) is projective iff all nodes occurring between i and j are dominated by i (where dominates is the transitive closure of the arc relation).

In the table, the languages are abbreviated to their first two letters. Language families are: Semitic, Isolate, Romance, Sino-Tibetan, Slavic, Germanic, Hellenic, Finno-Ugric, and Turkic. The type of the original annotation is either constituents plus (some) functions (c+f) or dependencies (d). For the training data, the number of words and sentences are given in multiples of thousands, and the average length of a sentence in words (including punctuation tokens). The following rows contain information about whether lemmas are available, the number of coarse- and fine-grained part-of-speech tags, the number of feature components, and the number of dependency labels. Then information is given on how many different dependency labels can co-occur with HEAD=0, the percentage of HEAD=0 dependencies, and the percentage of heads preceding (left) or succeeding (right) a token (giving an indication of whether a language is predominantly head-initial or head-final). This is followed by the average number of HEAD=0 dependencies per sentence and the percentage of non-projective arcs and sentences. The last two rows show whether punctuation tokens are attached as dependents of other tokens (A=Always, S=Sometimes) and specify the number of dependency labels that exist for punctuation tokens. Note that punctuation is defined as any token belonging to the UTF-8 category of punctuation. This means, for example, that any token having an underscore in the FORM field (which happens for word-internal IGs in Turkish) is also counted as punctuation here.

For the test sets, the number of words and sentences as well as the ratio of words per sentence are listed, followed by the percentage of new words and lemmas (if applicable). For the domain adaptation sets, the percentage of new words is computed with regard to the training set (Penn Treebank).

4 Submissions and Results

As already stated in the introduction, test runs were submitted for twenty-three systems in the multilingual track, and ten systems in the domain adaptation track (six of which also participated in the multilingual track). In the result tables below, systems are identified by the last name of the team member listed first when test runs were uploaded for evaluation. In general, this name is also the first author of a paper describing the system in the proceedings, but there are a few exceptions and complications. First of all, for four out of twenty-seven systems, no paper was submitted to the proceedings. This is the case for the systems of Jia, Maes et al., Nash, and Zeman, which is indicated by the fact that these names appear in italics in all result tables. Secondly, two teams submitted two systems each, which are described in a single paper by each team. Thus, the systems called "Nilsson" and "Hall, J." are both described in Hall et al. (2007a), while the systems called "Duan (1)" and "Duan (2)" are both described in Duan et al. (2007). Finally, please pay attention to the fact that there are two teams, where the first author's last name is Hall. Therefore, we use "Hall, J." and "Hall, K.", to disambiguate between the teams involving Johan Hall (Hall et al., 2007a) and Keith Hall (Hall et al., 2007b), respectively.

Tables 2 and 3 give the scores for the multilingual track in the CoNLL 2007 shared task. The Average column contains the average score for all ten languages, which determines the ranking in this track. Table 4 presents the results for the domain adaptation track, where the ranking is determined based on the PCHEM results only, since the CHILDES data set was optional. Note also that there are no labeled

Team	Average	Arabic	Basque	Catalan	Chinese	Czech	English	Greek	Hungarian	Italian	Turkish
Nilsson	80.32(1)	76.52(1)	76.94(1)	88.70(1)	75.82(15)	77.98(3)	88.11(5)	74.65(2)	80.27(1)	84.40(1)	79.79(2)
Nakagawa	80.29(2)	75.08(2)	72.56(7)	87.90(3)	83.84(2)	80.19(1)	88.41(3)	76.31(1)	76.74(8)	83.61(3)	78.22(5)
Titov	79.90(3)	74.12(6)	75.49(3)	87.40(6)	82.14(7)	77.94(4)	88.39(4)	73.52(10)	77.94(4)	82.26(6)	79.81(1)
Sagae	79.90(4)	74.71(4)	74.64(6)	88.16(2)	84.69(1)	74.83(8)	89.01(2)	73.58(8)	79.53(2)	83.91(2)	75.91(10)
Hall, J.	79.80(5)*	74.75(3)	74.99(5)	87.74(4)	83.51(3)	77.22(6)	85.81(12)	74.21(6)	78.09(3)	82.48(5)	79.24(3)
Carreras	79.09(6)*	70.20(11)	75.75(2)	87.60(5)	80.86(10)	78.60(2)	89.61(1)	73.56(9)	75.42(9)	83.46(4)	75.85(11)
Attardi	78.27(7)	72.66(8)	69.48(12)	86.86(7)	81.50(8)	77.37(5)	85.85(10)	73.92(7)	76.81(7)	81.34(8)	76.87(7)
Chen	78.06(8)	74.65(5)	72.39(8)	86.66(8)	81.24(9)	73.69(10)	83.81(13)	74.42(3)	75.34(10)	82.04(7)	76.31(9)
Duan (1)	77.70(9)*	69.91(13)	71.26(9)	84.95(10)	82.58(6)	75.34(7)	85.83(11)	74.29(4)	77.06(5)	80.75(9)	75.03(12)
Hall, K.	76.91(10)*	73.40(7)	69.81(11)	82.38(14)	82.77(4)	72.27(12)	81.93(15)	74.21(5)	74.20(11)	80.69(10)	77.42(6)
Schiehlen	76.18(11)	70.08(12)	66.77(14)	85.75(9)	80.04(11)	73.86(9)	86.21(9)	72.29(12)	73.90(12)	80.46(11)	72.48(15)
Johansson	75.78(12)*	71.76(9)	75.08(4)	83.33(12)	76.30(14)	70.98(13)	80.29(17)	72.77(11)	71.31(13)	77.55(14)	78.46(4)
Mannem	74.54(13)*	71.55(10)	65.64(15)	84.47(11)	73.76(17)	70.68(14)	81.55(16)	71.69(13)	70.94(14)	78.67(13)	76.42(8)
Wu	73.02(14)*	66.16(14)	70.71(10)	81.44(15)	74.69(16)	66.72(16)	79.49(18)	70.63(14)	69.08(15)	78.79(12)	72.52(14)
Nguyen	72.53(15)*	63.58(16)	58.18(17)	83.23(13)	79.77(12)	72.54(11)	86.73(6)	70.42(15)	68.12(17)	75.06(16)	67.63(17)
Maes	70.66(16)*	65.12(15)	69.05(13)	79.21(16)	70.97(18)	67.38(15)	69.68(21)	68.59(16)	68.93(16)	73.63(18)	74.03(13)
Canisius	66.99(17)*	59.13(18)	63.17(16)	75.44(17)	70.45(19)	56.14(17)	77.27(19)	60.35(18)	64.31(19)	75.57(15)	68.09(16)
Jia	63.00(18)*	63.37(17)	57.61(18)	23.35(20)	76.36(13)	54.95(18)	82.93(14)	65.45(17)	66.61(18)	74.65(17)	64.68(18)
Zeman	54.87(19)	46.06(20)	50.61(20)	62.94(19)	54.49(20)	50.21(20)	53.59(22)	55.29(19)	55.24(20)	62.13(19)	58.10(19)
Marinov	54.55(20)*	54.00(19)	51.24(19)	69.42(18)	49.87(21)	53.47(19)	52.11(23)	54.33(20)	44.47(21)	59.75(20)	56.88(20)
Duan (2)	24.62(21)*				82.64(5)		86.69(7)		76.89(6)		
Nash	8.65(22)*						86.49(8)				
Shimizu	7.20(23)						72.02(20)				

Table 2: Labeled attachment score (LAS) for the multilingual track in the CoNLL 2007 shared task. Teams are denoted by the last name of their first member, with italics indicating that there is no corresponding paper in the proceedings. The number in parentheses next to each score gives the rank. A star next to a score in the Average column indicates a statistically significant difference with the next lower rank.

Team	Average	Arabic	Basque	Catalan	Chinese	Czech	English	Greek	Hungarian	Italian	Turkish
Nakagawa	86.55(1)*	86.09(1)	81.04(5)	92.86(4)	88.88(2)	86.28(1)	90.13(2)	84.08(1)	82.49(3)	87.91(1)	85.77(3)
Nilsson	85.71(2)	85.81(2)	82.84(1)	93.12(3)	84.52(12)	83.59(4)	88.93(5)	81.22(4)	83.55(1)	87.77(2)	85.77(2)
Titov	85.62(3)	83.18(7)	81.93(2)	93.40(1)	87.91(4)	84.19(3)	89.73(4)	81.20(5)	82.18(4)	86.26(6)	86.22(1)
Sagae	85.29(4)*	84.04(4)	81.19(3)	93.34(2)	88.94(1)	81.27(8)	89.87(3)	80.37(11)	83.51(2)	87.68(3)	82.72(9)
Carreras	84.79(5)	81.48(10)	81.11(4)	92.46(5)	86.20(9)	85.16(2)	90.63(1)	81.37(3)	79.92(9)	87.19(4)	82.41(10)
Hall, J.	84.74(6)*	84.21(3)	80.61(6)	92.20(6)	87.60(5)	82.35(6)	86.77(12)	80.66(9)	81.71(6)	86.26(5)	85.04(5)
Attardi	83.96(7)*	82.53(8)	76.88(11)	91.41(7)	86.73(8)	83.40(5)	86.99(10)	80.75(8)	81.81(5)	85.54(8)	83.56(7)
Chen	83.22(8)	83.49(5)	78.65(8)	90.87(8)	85.91(10)	80.14(11)	84.91(13)	81.16(6)	79.25(11)	85.91(7)	81.92(12)
Hall, K.	83.08(9)	83.45(6)	78.55(9)	87.80(15)	87.91(3)	78.47(12)	83.21(15)	82.04(2)	79.34(10)	84.81(9)	85.18(4)
Duan (1)	82.77(10)	79.04(13)	77.59(10)	89.71(12)	86.88(7)	80.82(10)	86.97(11)	80.77(7)	80.66(7)	84.20(11)	81.03(13)
Schiehlen	82.42(11)*	81.07(11)	73.30(14)	90.79(10)	85.45(11)	81.73(7)	88.91(6)	80.47(10)	78.61(12)	84.54(10)	79.33(15)
Johansson	81.13(12)*	80.91(12)	80.43(7)	88.34(13)	81.30(15)	77.39(13)	81.43(18)	79.58(12)	75.53(15)	81.55(15)	84.80(6)
Mannem	80.30(13)	81.56(9)	72.88(15)	89.81(11)	78.84(17)	77.20(14)	82.81(16)	78.89(13)	75.39(16)	82.91(12)	82.74(8)
Nguyen	80.00(14)*	73.46(18)	69.15(18)	88.12(14)	84.05(13)	80.91(9)	88.01(7)	77.56(15)	78.13(13)	80.40(16)	80.19(14)
Jia	78.46(15)	74.20(17)	70.24(16)	90.83(9)	83.39(14)	70.41(18)	84.37(14)	75.65(16)	77.19(14)	82.36(14)	75.96(17)
Wu	78.44(16)*	77.05(14)	75.77(12)	85.85(16)	79.71(16)	73.07(16)	81.69(17)	78.12(14)	72.39(18)	82.57(13)	78.15(16)
Maes	76.60(17)*	75.47(16)	75.27(13)	84.35(17)	76.57(18)	74.03(15)	71.62(21)	75.19(17)	72.93(17)	78.32(18)	82.21(11)
Canisius	74.83(18)*	76.89(15)	70.17(17)	81.64(18)	74.81(19)	72.12(17)	78.23(19)	72.46(18)	67.80(19)	79.08(17)	75.14(18)
Zeman	62.02(19)*	58.55(20)	57.42(20)	68.50(20)	62.93(20)	59.19(20)	58.33(22)	62.89(19)	59.78(20)	68.27(19)	64.30(19)
Marinov	60.83(20)*	64.27(19)	58.55(19)	74.22(19)	56.09(21)	59.57(19)	54.33(23)	61.18(20)	50.39(21)	65.52(20)	64.13(20)
Duan (2)	25.53(21)*				86.94(6)		87.87(8)		80.53(8)		
Nash	8.77(22)*						87.71(9)				
Shimizu	7.79(23)						77.91(20)				

Table 3: Unlabeled attachment scores (UAS) for the multilingual track in the CoNLL 2007 shared task. Teams are denoted by the last name of their first member, with italics indicating that there is no corresponding paper in the proceedings. The number in parentheses next to each score gives the rank. A star next to a score in the Average column indicates a statistically significant difference with the next lower rank.

	LAS		UAS			
Team	PCHEM-c	PCHEM-o	PCHEM-c	PCHEM-o	CHILDES-c	CHILDES-o
Sagae	81.06(1)		83.42(1)			
Attardi	80.40(2)		83.08(3)		58.67(3)	
Dredze	80.22(3)		83.38(2)		61.37(1)	
Nguyen	79.50(4)*		82.04(4)*			
Jia	76.48(5)*		78.92(5)*		57.43(5)	
Bick	71.81(6)*	78.48(1)*	74.71(6)*	81.62(1)*	58.07(4)	62.49(1)
Shimizu	64.15(7)*	63.49(2)	71.25(7)*	70.01(2)*		
Zeman	50.61(8)		54.57(8)		58.89(2)	
Schneider		63.01(3)*		66.53(3)*		60.27(2)
Watson		55.47(4)		62.79(4)		45.61(3)
Wu					52.89(6)	

Table 4: Labeled (LAS) and unlabeled (UAS) attachment scores for the closed (-c) and open (-o) classes of the domain adaptation track in the CoNLL 2007 shared task. Teams are denoted by the last name of their first member, with italics indicating that there is no corresponding paper in the proceedings. The number in parentheses next to each score gives the rank. A star next to a score in the PCHEM columns indicates a statistically significant difference with the next lower rank.

attachment scores for the CHILDES data set, for reasons explained in section 3.2. The number in parentheses next to each score gives the rank. A star next to a score indicates that the difference with the next lower rank is significant at the 5% level using a z-test for proportions. A more complete presentation of the results, including the significance results for all the tasks and their p-values, can be found on the shared task website.[4]

Looking first at the results in the multilingual track, we note that there are a number of systems performing at almost the same level at the top of the ranking. For the average labeled attachment score, the difference between the top score (Nilsson) and the fifth score (Hall, J.) is no more than half a percentage point, and there are generally very few significant differences among the five or six best systems, regardless of whether we consider labeled or unlabeled attachment score. For the closed class of the domain adaptation track, we see a very similar pattern, with the top system (Sagae) being followed very closely by two other systems. For the open class, the results are more spread out, but then there are very few results in this class. It is also worth noting that the top scores in the closed class, somewhat unexpectedly, are higher than the top scores in the open class. But before we proceed to a more detailed analysis of the results (section 6), we will make an attempt to characterize the approaches represented by the different systems.

5 Approaches

In this section we give an overview of the models, inference methods, and learning methods used in the participating systems. For obvious reasons the discussion is limited to systems that are described by a paper in the proceedings. But instead of describing the systems one by one, we focus on the basic methodological building blocks that are often found in several systems although in different combinations. For descriptions of the individual systems, we refer to the respective papers in the proceedings.

Section 5.1 is devoted to system architectures. We then describe the two main paradigms for learning and inference, in this year's shared task as well as in last year's, which we call *transition-based* parsers (section 5.2) and *graph-based* parsers (section 5.3), adopting the terminology of McDonald and Nivre (2007).[5] Finally, we give an overview of the domain adaptation methods that were used (section 5.4).

[4] http://nextens.uvt.nl/depparse-wiki/AllScores

[5] This distinction roughly corresponds to the distinction made by Buchholz and Marsi (2006) between ”stepwise” and “all-pairs” approaches.

5.1 Architectures

Most systems perform some amount of pre- and post-processing, making the actual parsing component part of a sequential workflow of varying length and complexity. For example, most transition-based parsers can only build projective dependency graphs. For languages with non-projective dependencies, graphs therefore need to be projectivized for training and deprojectivized for testing (Hall et al., 2007a; Johansson and Nugues, 2007b; Titov and Henderson, 2007).

Instead of assigning HEAD and DEPREL in a single step, some systems use a two-stage approach for attaching and labeling dependencies (Chen et al., 2007; Dredze et al., 2007). In the first step unlabeled dependencies are generated, in the second step these are labeled. This is particularly helpful for factored parsing models, in which label decisions cannot be easily conditioned on larger parts of the structure due to the increased complexity of inference. One system (Hall et al., 2007b) extends this two-stage approach to a three-stage architecture where the parser and labeler generate an n-best list of parses which in turn is reranked.[6]

In ensemble-based systems several base parsers provide parsing decisions, which are added together for a combined score for each potential dependency arc. The tree that maximizes the sum of these combined scores is taken as the final output parse. This technique is used by Sagae and Tsujii (2007) and in the Nilsson system (Hall et al., 2007a). It is worth noting that both these systems combine transition-based base parsers with a graph-based method for parser combination, as first described by Sagae and Lavie (2006).

Data-driven grammar-based parsers, such as Bick (2007), Schneider et al. (2007), and Watson and Briscoe (2007), need pre- and post-processing in order to map the dependency graphs provided as training data to a format compatible with the grammar used, and vice versa.

5.2 Transition-Based Parsers

Transition-based parsers build dependency graphs by performing sequences of actions, or transitions. Both learning and inference is conceptualized in terms of predicting the correct transition based on the current parser state and/or history. We can further subclassify parsers with respect to the model (or transition system) they adopt, the inference method they use, and the learning method they employ.

5.2.1 Models

The most common model for transition-based parsers is one inspired by shift-reduce parsing, where a parser state contains a stack of partially processed tokens and a queue of remaining input tokens, and where transitions add dependency arcs and perform stack and queue operations. This type of model is used by the majority of transition-based parsers (Attardi et al., 2007; Duan et al., 2007; Hall et al., 2007a; Johansson and Nugues, 2007b; Mannem, 2007; Titov and Henderson, 2007; Wu et al., 2007). Sometimes it is combined with an explicit probability model for transition sequences, which may be conditional (Duan et al., 2007) or generative (Titov and Henderson, 2007).

An alternative model is based on the list-based parsing algorithm described by Covington (2001), which iterates over the input tokens in a sequential manner and evaluates for each preceding token whether it can be linked to the current token or not. This model is used by Marinov (2007) and in component parsers of the Nilsson ensemble system (Hall et al., 2007a). Finally, two systems use models based on LR parsing (Sagae and Tsujii, 2007; Watson and Briscoe, 2007).

5.2.2 Inference

The most common inference technique in transition-based dependency parsing is greedy deterministic search, guided by a classifier for predicting the next transition given the current parser state and history, processing the tokens of the sentence in sequential left-to-right order[7] (Hall et al., 2007a; Mannem, 2007; Marinov, 2007; Wu et al., 2007). Optionally multiple passes over the input are conducted until no tokens are left unattached (Attardi et al., 2007).

As an alternative to deterministic parsing, several parsers use probabilistic models and maintain a heap or beam of partial transition sequences in order to pick the most probable one at the end of the sentence

[6] They also flip the order of the labeler and the reranker.

[7] For diversity in parser ensembles, right-to-left parsers are also used.

(Duan et al., 2007; Johansson and Nugues, 2007b; Sagae and Tsujii, 2007; Titov and Henderson, 2007).

One system uses as part of their parsing pipeline a "neighbor-parser" that attaches adjacent words and a "root-parser" that identifies the root word(s) of a sentence (Wu et al., 2007). In the case of grammar-based parsers, a classifier is used to disambiguate in cases where the grammar leaves some ambiguity (Schneider et al., 2007; Watson and Briscoe, 2007)

5.2.3 Learning

Transition-based parsers either maintain a classifier that predicts the next transition or a global probabilistic model that scores a complete parse. To train these classifiers and probabilitistic models several approaches were used: SVMs (Duan et al., 2007; Hall et al., 2007a; Sagae and Tsujii, 2007), modified finite Newton SVMs (Wu et al., 2007), maximum entropy models (Sagae and Tsujii, 2007), multiclass averaged perceptron (Attardi et al., 2007) and maximum likelihood estimation (Watson and Briscoe, 2007).

In order to calculate a global score or probability for a transition sequence, two systems used a Markov chain approach (Duan et al., 2007; Sagae and Tsujii, 2007). Here probabilities from the output of a classifier are multiplied over the whole sequence of actions. This results in a locally normalized model. Two other entries used MIRA (Mannem, 2007) or online passive-aggressive learning (Johansson and Nugues, 2007b) to train a globally normalized model. Titov and Henderson (2007) used an incremental sigmoid Bayesian network to model the probability of a transition sequence and estimated model parameters using neural network learning.

5.3 Graph-Based Parsers

While transition-based parsers use training data to learn a process for deriving dependency graphs, graph-based parsers learn a model of what it means to be a good dependency graph given an input sentence. They define a scoring or probability function over the set of possible parses. At learning time they estimate parameters of this function; at parsing time they search for the graph that maximizes this function. These parsers mainly differ in the type and structure of the scoring function (model), the search algorithm that finds the best parse (inference), and the method to estimate the function's parameters (learning).

5.3.1 Models

The simplest type of model is based on a sum of local attachment scores, which themselves are calculated based on the dot product of a weight vector and a feature representation of the attachment. This type of scoring function is often referred to as a first-order model.[8] Several systems participating in this year's shared task used first-order models (Schiehlen and Spranger, 2007; Nguyen et al., 2007; Shimizu and Nakagawa, 2007; Hall et al., 2007b). Canisius and Tjong Kim Sang (2007) cast the same type of arc-based factorization as a weighted constraint satisfaction problem.

Carreras (2007) extends the first-order model to incorporate a sum over scores for pairs of adjacent arcs in the tree, yielding a second-order model. In contrast to previous work where this was constrained to sibling relations of the dependent (McDonald and Pereira, 2006), here head-grandchild relations can be taken into account.

In all of the above cases the scoring function is decomposed into functions that score local properties (arcs, pairs of adjacent arcs) of the graph. By contrast, the model of Nakagawa (2007) considers global properties of the graph that can take multiple arcs into account, such as multiple siblings and children of a node.

5.3.2 Inference

Searching for the highest scoring graph (usually a tree) in a model depends on the factorization chosen and whether we are looking for projective or non-projective trees. Maximum spanning tree algorithms can be used for finding the highest scoring non-projective tree in a first-order model (Hall et al., 2007b; Nguyen et al., 2007; Canisius and Tjong Kim Sang, 2007; Shimizu and Nakagawa, 2007), while Eisner's dynamic programming algorithm solves the problem for a first-order factorization in the projective case (Schiehlen and Spranger, 2007). Carreras (2007) employs his own extension of Eisner's algorithm for the case of projective trees and second-order models that include head-grandparent relations.

[8] It is also known as an edge-factored model.

The methods presented above are mostly efficient and always exact. However, for models that take global properties of the tree into account, they cannot be applied. Instead Nakagawa (2007) uses Gibbs sampling to obtain marginal probabilities of arcs being included in the tree using his global model and then applies a maximum spanning tree algorithm to maximize the sum of the logs of these marginals and return a valid cycle-free parse.

5.3.3 Learning

Most of the graph-based parsers were trained using an online inference-based method such as passive-aggressive learning (Nguyen et al., 2007; Schiehlen and Spranger, 2007), averaged perceptron (Carreras, 2007), or MIRA (Shimizu and Nakagawa, 2007), while some systems instead used methods based on maximum conditional likelihood (Nakagawa, 2007; Hall et al., 2007b).

5.4 Domain Adaptation

5.4.1 Feature-Based Approaches

One way of adapting a learner to a new domain without using any unlabeled data is to only include features that are expected to transfer well (Dredze et al., 2007). In structural correspondence learning a transformation from features in the source domain to features of the target domain is learnt (Shimizu and Nakagawa, 2007). The original source features along with their transformed versions are then used to train a discriminative parser.

5.4.2 Ensemble-Based Approaches

Dredze et al. (2007) trained a diverse set of parsers in order to improve cross-domain performance by incorporating their predictions as features for another classifier. Similarly, two parsers trained with different learners and search directions were used in the co-learning approach of Sagae and Tsujii (2007). Unlabeled target data was processed with both parsers. Sentences that both parsers agreed on were then added to the original training data. This combined data set served as training data for one of the original parsers to produce the final system. In a similar fashion, Watson and Briscoe (2007) used a variant of self-training to make use of the unlabeled target data.

5.4.3 Other Approaches

Attardi et al. (2007) learnt tree revision rules for the target domain by first parsing unlabeled target data using a strong parser; this data was then combined with labeled source data; a weak parser was applied to this new dataset; finally tree correction rules are collected based on the mistakes of the weak parser with respect to the gold data and the output of the strong parser.

Another technique used was to filter sentences of the out-of-domain corpus based on their similarity to the target domain, as predicted by a classifier (Dredze et al., 2007). Only if a sentence was judged similar to target domain sentences was it included in the training set.

Bick (2007) used a hybrid approach, where a data-driven parser trained on the labeled training data was given access to the output of a Constraint Grammar parser for English run on the same data. Finally, Schneider et al. (2007) learnt collocations and relational nouns from the unlabeled target data and used these in their parsing algorithm.

6 Analysis

Having discussed the major approaches taken in the two tracks of the shared task, we will now return to the test results. For the multilingual track, we compare results across data sets and across systems, and report results from a parser combination experiment involving all the participating systems (section 6.1). For the domain adaptation track, we sum up the most important findings from the test results (section 6.2).

6.1 Multilingual Track

6.1.1 Across Data Sets

The average LAS over all systems varies from 68.07 for Basque to 80.95 for English. Top scores vary from 76.31 for Greek to 89.61 for English. In general, there is a good correlation between the top scores and the average scores. For Greek, Italian, and Turkish, the top score is closer to the average score than the average distance, while for Czech, the distance is higher. The languages that produced the most stable results in terms of system ranks with respect to LAS are Hungarian and Italian. For UAS, Catalan also falls into this group. The language that

Setup	Arabic	Chinese	Czech	Turkish
2006 without punctuation	66.9	90.0	80.2	65.7
2007 without punctuation	75.5	84.9	80.0	71.6
2006 with punctuation	67.0	90.0	80.2	73.8
2007 with punctuation	76.5	84.7	80.2	79.8

Table 5: A comparison of the LAS top scores from 2006 and 2007. Official scoring conditions in boldface. For Turkish, scores with punctuation also include word-internal dependencies.

produced the most unstable results with respect to LAS is Turkish.

In comparison to last year's languages, the languages involved in the multilingual track this year can be more easily separated into three classes with respect to top scores:

- Low (76.31–76.94):
  Arabic, Basque, Greek
- Medium (79.19–80.21):
  Czech, Hungarian, Turkish
- High (84.40–89.61):
  Catalan, Chinese, English, Italian

It is interesting to see that the classes are more easily definable via language characteristics than via characteristics of the data sets. The split goes across training set size, original data format (constituent vs. dependency), sentence length, percentage of unknown words, number of dependency labels, and ratio of (C)POSTAGS and dependency labels. The class with the highest top scores contains languages with a rather impoverished morphology. Medium scores are reached by the two agglutinative languages, Hungarian and Turkish, as well as by Czech. The most difficult languages are those that combine a relatively free word order with a high degree of inflection. Based on these characteristics, one would expect to find Czech in the last class. However, the Czech training set is four times the size of the training set for Arabic, which is the language with the largest training set of the difficult languages.

However, it would be wrong to assume that training set size alone is the deciding factor. A closer look at table 1 shows that while Basque and Greek in fact have small training data sets, so do Turkish and Italian. Another factor that may be associated with the above classification is the percentage of new words (PNW) in the test set. Thus, the expectation would be that the highly inflecting languages have a high PNW while the languages with little morphology have a low PNW. But again, there is no direct correspondence. Arabic, Basque, Catalan, English, and Greek agree with this assumption: Catalan and English have the smallest PNW, and Arabic, Basque, and Greek have a high PNW. But the PNW for Italian is higher than for Arabic and Greek, and this is also true for the percentage of new lemmas. Additionally, the highest PNW can be found in Hungarian and Turkish, which reach higher scores than Arabic, Basque, and Greek. These considerations suggest that highly inflected languages with (relatively) free word order need more training data, a hypothesis that will have to be investigated further.

There are four languages which were included in the shared tasks on multilingual dependency parsing both at CoNLL 2006 and at CoNLL 2007: Arabic, Chinese, Czech, and Turkish. For all four languages, the same treebanks were used, which allows a comparison of the results. However, in some cases the size of the training set changed, and at least one treebank, Turkish, underwent a thorough correction phase. Table 5 shows the top scores for LAS. Since the official scores excluded punctuation in 2006 but includes it in 2007, we give results both with and without punctuation for both years.

For Arabic and Turkish, we see a great improvement of approximately 9 and 6 percentage points. For Arabic, the number of tokens in the training set doubled, and the morphological annotation was made more informative. The combined effect of these changes can probably account for the substantial improvement in parsing accuracy. For Turkish, the training set grew in size as well, although only by 600 sentences, but part of the improvement for Turkish may also be due to continuing efforts in error correction and consistency checking. We see that the choice to include punctuation or not makes a large difference for the Turkish scores, since non-final IGs of a word are counted as punctuation (because they have the underscore character as their FORM value), which means that word-internal dependency links are included if punctuation is included.[9] However, regardless of whether we compare scores with or without punctuation, we see a genuine improvement of approximately 6 percentage points.

For Chinese, the same training set was used. Therefore, the drop from last year's top score to this year's is surprising. However, last year's top scoring system for Chinese (Riedel et al., 2006), which did not participate this year, had a score that was more than 3 percentage points higher than the second best system for Chinese. Thus, if we compare this year's results to the second best system, the difference is approximately 2 percentage points. This final difference may be attributed to the properties of the test sets. While last year's test set was taken from the treebank, this year's test set contains texts from other sources. The selection of the textual basis also significantly changed average sentence length: The Chinese training set has an average sentence length of 5.9. Last year's test set also had an average sentence length of 5.9. However, this year, the average sentence length is 7.5 tokens, which is a significant increase. Longer sentences are typically harder to parse due to the increased likelihood of ambiguous constructions.

Finally, we note that the performance for Czech is almost exactly the same as last year, despite the fact that the size of the training set has been reduced to approximately one third of last year's training set. It is likely that this in fact represents a relative improvement compared to last year's results.

6.1.2 Across Systems

The LAS over all languages ranges from 80.32 to 54.55. The comparison of the system ranks averaged over all languages with the ranks for single languages show considerably more variation than last year's systems. Buchholz and Marsi (2006) report that "[f]or most parsers, their ranking differs at most a few places from their overall ranking". This year, for all of the ten best performing systems with respect to LAS, there is at least one language for which their rank is at least 5 places different from their overall rank. The most extreme case is the top performing Nilsson system (Hall et al., 2007a), which reached rank 1 for five languages and rank 2 for two more languages. Their only outlier is for Chinese, where the system occupies rank 14, with a LAS approximately 9 percentage points below the top scoring system for Chinese (Sagae and Tsujii, 2007). However, Hall et al. (2007a) point out that the official results for Chinese contained a bug, and the true performance of their system was actually much higher. The greatest improvement of a system with respect to its average rank occurs for English, for which the system by Nguyen et al. (2007) improved from the average rank 15 to rank 6. Two more outliers can be observed in the system of Johansson and Nugues (2007b), which improves from its average rank 12 to rank 4 for Basque and Turkish. The authors attribute this high performance to their parser's good performance on small training sets. However, this hypothesis is contradicted by their results for Greek and Italian, the other two languages with small training sets. For these two languages, the system's rank is very close to its average rank.

6.1.3 An Experiment in System Combination

Having the outputs of many diverse dependency parsers for standard data sets opens up the interesting possibility of parser combination. To combine the outputs of each parser we used the method of Sagae and Lavie (2006). This technique assigns to each possible labeled dependency a weight that is equal to the number of systems that included the dependency in their output. This can be viewed as an arc-based voting scheme. Using these weights it is possible to search the space of possible dependency trees using directed maximum spanning tree algorithms (McDonald et al., 2005). The maximum spanning tree in this case is equal to the tree that on average contains the labeled dependencies that most systems voted for. It is worth noting that variants of this scheme were used in two of the participating

[9] The decision to include word-internal dependencies in this way can be debated on the grounds that they can be parsed deterministically. On the other hand, they typically correspond to regular dependencies captured by function words in other languages, which are often easy to parse as well. It is therefore unclear whether scores are more inflated by including word-internal dependencies or deflated by excluding them.

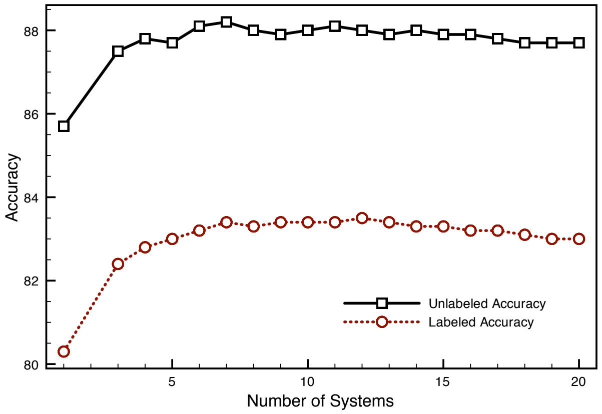


Figure 1: System Combination

systems, the Nilsson system (Hall et al., 2007a) and the system of Sagae and Tsujii (2007).

Figure 1 plots the labeled and unlabeled accuracies when combining an increasing number of systems. The data used in the plot was the output of all competing systems for every language in the multilingual track. The plot was constructed by sorting the systems based on their average labeled accuracy scores over all languages, and then incrementally adding each system in descending order.[10] We can see that both labeled and unlabeled accuracy are significantly increased, even when just the top three systems are included. Accuracy begins to degrade gracefully after about ten different parsers have been added. Furthermore, the accuracy never falls below the performance of the top three systems.

6.2 Domain Adaptation Track

For this task, the results are rather surprising. A look at the LAS and UAS for the chemical research abstracts shows that there are four closed systems that outperform the best scoring open system. The best system (Sagae and Tsujii, 2007) reaches an LAS of 81.06 (in comparison to their LAS of 89.01 for the English data set in the multilingual track). Considering that approximately one third of the words of the chemical test set are new, the results are noteworthy.

The next surprise is to be found in the relatively low UAS for the CHILDES data. At a first glance, this data set has all the characteristics of an easy set; the average sentence is short (12.9 words), and the percentage of new words is also small (6.10%). Despite these characteristics, the top UAS reaches 62.49 and is thus more than 10 percentage points below the top UAS for the chemical data set. One major reason for this is that auxiliary and main verb dependencies are annotated differently in the CHILDES data than in the WSJ training set. As a result of this discrepancy, participants were not required to submit results for the CHILDES data. The best performing system on the CHILDES corpus is an open system (Bick, 2007), but the distance to the top closed system is approximately 1 percentage point. In this domain, it seems more feasible to use general language resources than for the chemical domain. However, the results prove that the extra effort may be unnecessary.

7 Conclusion

Two years of dependency parsing in the CoNLL shared task has brought an enormous boost to the development of dependency parsers for multiple languages (and to some extent for multiple domains). But even though nineteen languages have been covered by almost as many different parsing and learning approaches, we still have only vague ideas about the strengths and weaknesses of different methods for languages with different typological characteristics. Increasing our knowledge of the multi-causal relationship between language structure, annotation scheme, and parsing and learning methods probably remains the most important direction for future research in this area. The outputs of all systems for all data sets from the two shared tasks are freely available for research and constitute a potential gold mine for comparative error analysis across languages and systems.

For domain adaptation we have barely scratched the surface so far. But overcoming the bottleneck of limited annotated resources for specialized domains will be as important for the deployment of human language technology as being able to handle multiple languages in the future. One result from the domain adaptation track that may seem surprising at first is the fact that closed class systems outperformed open class systems on the chemical abstracts. However, it seems that the major problem in

[10] The reason that there is no data point for two parsers is that the simple voting scheme adopted only makes sense with at least three parsers voting.

adapting pre-existing parsers to the new domain was not the domain as such but the mapping from the native output of the parser to the kind of annotation provided in the shared task data sets. Thus, finding ways of reusing already invested development efforts by adapting the outputs of existing systems to new requirements, without substantial loss in accuracy, seems to be another line of research that may be worth pursuing.

Acknowledgments

First and foremost, we want to thank all the people and organizations that generously provided us with treebank data and helped us prepare the data sets and without whom the shared task would have been literally impossible: Otakar Smrz, Charles University, and the LDC (Arabic); Maxux Aranzabe, Kepa Bengoetxea, Larraitz Uria, Koldo Gojenola, and the University of the Basque Country (Basque); Ma. Antònia Martí Antonín, Lluís Màrquez, Manuel Bertran, Mariona Taulé, Difda Monterde, Eli Comelles, and CLiC-UB (Catalan); Shih-Min Li, Keh-Jiann Chen, Yu-Ming Hsieh, and Academia Sinica (Chinese); Jan Hajič, Zdenek Zabokrtsky, Charles University, and the LDC (Czech); Brian MacWhinney, Eric Davis, the CHILDES project, the Penn BioIE project, and the LDC (English); Prokopis Prokopidis and ILSP (Greek); Csirik János and Zoltán Alexin (Hungarian); Giuseppe Attardi, Simonetta Montemagni, Maria Simi, Isidoro Barraco, Patrizia Topi, Kiril Ribarov, Alessandro Lenci, Nicoletta Calzolari, ILC, and ELRA (Italian); Gülşen Eryiğit, Kemal Oflazer, and Ruket Çakıcı (Turkish).

Secondly, we want to thank the organizers of last year's shared task, Sabine Buchholz, Amit Dubey, Erwin Marsi, and Yuval Krymolowski, who solved all the really hard problems for us and answered all our questions, as well as our colleagues who helped review papers: Jason Baldridge, Sabine Buchholz, James Clarke, Gülşen Eryiğit, Kilian Evang, Julia Hockenmaier, Yuval Krymolowski, Erwin Marsi, Beáta Megyesi, Yannick Versley, and Alexander Yeh. Special thanks to Bertjan Busser and Erwin Marsi for help with the CoNLL shared task website and many other things, and to Richard Johansson for letting us use his conversion tool for English.

Thirdly, we want to thank the program chairs for EMNLP-CoNLL 2007, Jason Eisner and Taku Kudo, the publications chair, Eric Ringger, the SIGNLL officers, Antal van den Bosch, Hwee Tou Ng, and Erik Tjong Kim Sang, and members of the LDC staff, Tony Castelletto and Ilya Ahtaridis, for great cooperation and support.

Finally, we want to thank the following people, who in different ways assisted us in the organization of the CoNLL 2007 shared task: Giuseppe Attardi, Eckhard Bick, Matthias Buch-Kromann, Xavier Carreras, Tomaz Erjavec, Svetoslav Marinov, Wolfgang Menzel, Xue Nianwen, Gertjan van Noord, Petya Osenova, Florian Schiel, Kiril Simov, Zdenka Uresova, and Heike Zinsmeister.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*.

G. Attardi, F. Dell'Orletta, M. Simi, A. Chanev, and M. Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using desr. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

E. Bick. 2007. Hybrid ways to improve domain independence in an ML dependency parser. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, pages 103–127.

R. Brown. 1973. *A First Language: The Early Stages*. Harvard University Press.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL)*.

S. Canisius and E. Tjong Kim Sang. 2007. A constraint satisfaction approach to dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

X. Carreras. 2007. Experiments with a high-order projective dependency parser. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

C. Chelba and A. Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (2003), chapter 13, pages 231–248.

W. Chen, Y. Zhang, and H. Isahara. 2007. A two-stage parser for multilingual dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*.

M. A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proc. of the 39th Annual ACM Southeast Conf.*

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

H. Daumé and D. Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

M. Dredze, J. Blitzer, P. P. Talukdar, K. Ganchev, J. Graca, and F. Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

X. Duan, J. Zhao, and B. Xu. 2007. Probabilistic parsing action models for multi-lingual dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

G. Eryiğit. 2007. ITU validation set for Metu-Sabancı Turkish Treebank. URL: http://www3.itu.edu.tr/∼gulsenc/papers/validationset.pdf.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, A. Luo, N. Nicolov, and S. Roukos. 2004. A statisical model for multilingual entity detection and tracking. In *Proc. of the Human Language Technology Conf. and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

D. Gildea. 2001. Corpus variation and parser performance. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*.

J. Hall, J. Nilsson, J. Nivre, G. Eryiğit, B. Megyesi, M. Nilsson, and M. Saers. 2007a. Single malt or blended? A study in multilingual parser optimization. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

K. Hall, J. Havelka, and D. Smith. 2007b. Log-linear models of non-projective trees, k-best MST parsing and tree-ranking. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

R. Johansson and P. Nugues. 2007a. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conf. on Computational Linguistics (NODALIDA)*.

R. Johansson and P. Nugues. 2007b. Incremental dependency parsing using online learning. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of the Sixth Conf. on Computational Language Learning (CoNLL)*.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conf. and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

B. MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum.

P. R. Mannem. 2007. Online learning for deterministic dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

S. Marinov. 2007. Covington variations. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of the 11th Conf. of the European Chapter of the Association for Computational Linguistics (EACL)*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Human Language Technology Conf. and the Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

T. Nakagawa. 2007. Multilingual dependency parsing using Gibbs sampling. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

L.-M. Nguyen, T.-P. Nguyen, and A. Shimazu. 2007. A multilingual dependency analysis system using online passive-aggressive learning. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryiğit, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*.

S. Riedel, R. Çakıcı, and I. Meza-Ruiz. 2006. Multilingual dependency parsing with incremental integer linear programming. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL)*.

B. Roark and M. Bacchiani. 2003. Supervised and unsupervised PCFG adaptation to novel domains. In *Proc. of the Human Language Technology Conf. and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proc. of the Human Language Technology Conf. of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL)*.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

M. Schiehlen and Kristina Spranger. 2007. Global learning of labelled dependency trees. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

G. Schneider, K. Kaljurand, F. Rinaldi, and T. Kuhn. 2007. Pro3Gres parser in the CoNLL domain adaptation shared task. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

N. Shimizu and H. Nakagawa. 2007. Structural correspondence learning for dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

I. Titov and J. Henderson. 2006. Porting statistical parsers with data-defined kernels. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL)*.

I. Titov and J. Henderson. 2007. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

R. Watson and T. Briscoe. 2007. Adapting the RASP system for the CoNLL07 domain-adaptation task. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

Y.-C. Wu, J.-C. Yang, and Y.-S. Lee. 2007. Multilingual deterministic dependency parsing framework using modified finite Newton method support vector machines. In *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. 8th International Workshop on Parsing Technologies (IWPT)*.

Single Malt or Blended? A Study in Multilingual Parser Optimization

Johan Hall[*] **Jens Nilsson**[*] **Joakim Nivre**[*†]
Gülşen Eryiğit[‡] **Beáta Megyesi**[†] **Mattias Nilsson**[†] **Markus Saers**[†]

[*]Växjö University, School of Mathematics and Systems Engineering
E-mail: *firstname.lastname*@vxu.se

[†]Uppsala University, Dept. of Linguistics and Philology
E-mail: *firstname.lastname*@lingfil.uu.se

[‡]Istanbul Technical University, Computer Engineering Dept.
E-mail: gulsen.cebiroglu@itu.edu.tr

Abstract

We describe a two-stage optimization of the MaltParser system for the ten languages in the multilingual track of the CoNLL 2007 shared task on dependency parsing. The first stage consists in tuning a single-parser system for each language by optimizing parameters of the parsing algorithm, the feature model, and the learning algorithm. The second stage consists in building an ensemble system that combines six different parsing strategies, extrapolating from the optimal parameters settings for each language. When evaluated on the official test sets, the ensemble system significantly outperforms the single-parser system and achieves the highest average labeled attachment score.

1 Introduction

In the multilingual track of the CoNLL 2007 shared task on dependency parsing, a single parser must be trained to handle data from ten different languages: Arabic (Hajič et al., 2004), Basque (Aduriz et al., 2003), Catalan, (Martí et al., 2007), Chinese (Chen et al., 2003), Czech (Böhmová et al., 2003), English (Marcus et al., 1993; Johansson and Nugues, 2007), Greek (Prokopidis et al., 2005), Hungarian (Csendes et al., 2005), Italian (Montemagni et al., 2003), and Turkish (Oflazer et al., 2003).[1] Our contribution is a study in multilingual parser optimization using the freely available MaltParser system, which performs deterministic, classifier-based parsing with history-based feature models and discriminative learning, and which was one of the top performing systems in the CoNLL 2006 shared task (Nivre et al., 2006).

In order to maximize parsing accuracy, optimization has been carried out in two stages, leading to two different, but related parsers. The first of these is a single-parser system, similar to the one described in Nivre et al. (2006), which parses a sentence deterministically in a single left-to-right pass, with post-processing to recover non-projective dependencies, and where the parameters of the MaltParser system have been tuned for each language separately. We call this system Single Malt, to emphasize the fact that it consists of a single instance of MaltParser. The second parser is an ensemble system, which combines the output of six deterministic parsers, each of which is a variation of the Single Malt parser with parameter settings extrapolated from the first stage of optimization. It seems very natural to call this system Blended.

Section 2 summarizes the work done to optimize the Single Malt parser, while section 3 explains how the Blended parser was constructed from the Single Malt parser. Section 4 gives a brief analysis of the experimental results, and section 5 concludes.

2 The Single Malt Parser

The parameters available in the MaltParser system can be divided into three groups: parsing algorithm parameters, feature model parameters, and learning algorithm parameters.[2] Our overall optimization

[1] For more information about the task and the data sets, see Nivre et al. (2007).

[2] For a complete documentation of these parameters, see http://w3.msi.vxu.se/users/nivre/research/MaltParser.html.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 933–939,
Prague, June 2007. ©2007 Association for Computational Linguistics

strategy for the Single Malt parser was as follows:

1. Define a good baseline system with the same parameter settings for all languages.

2. Tune parsing algorithm parameters once and for all for each language (with baseline settings for feature model and learning algorithm parameters).

3. Optimize feature model and learning algorithm parameters in an interleaved fashion for each language.

We used nine-fold cross-validation on 90% of the training data for all languages with a training set size smaller than 300,000 tokens and an 80%–10% train-devtest split for the remaining languages (Catalan, Chinese, Czech, English). The remaining 10% of the data was in both cases saved for a final dry run, where the parser was trained on 90% of the data for each language and tested on the remaining (fresh) 10%. We consistently used the labeled attachment score (LAS) as the single optimization criterion.

Below we describe the most important parameters in each group, define baseline settings, and report notable improvements for different languages during development. The improvements for each language from step 1 (baseline) to step 2 (parsing algorithm) and step 3 (feature model and learning algorithm) can be tracked in table 1.[3]

2.1 Parsing Algorithm

MaltParser implements several parsing algorithms, but for the Single Malt system we stick to the one used by Nivre et al. (2006), which performs labeled projective dependency parsing in linear time, using a stack to store partially processed tokens and an input queue of remaining tokens. There are three basic parameters that can be varied for this algorithm:

1. **Arc order:** The baseline algorithm is *arc-eager*, in the sense that right dependents are attached to their head as soon as possible, but there is also an *arc-standard* version, where the attachment of right dependents has to be postponed until they have found all their own dependents. The arc-standard order was found to improve parsing accuracy for Chinese, while the arc-eager order was maintained for all other languages.

2. **Stack initialization:** In the baseline version the parser is initialized with an artificial root node (with token id 0) on the stack, so that arcs originating from the root can be added explicitly during parsing. But it is also possible to initialize the parser with an empty stack, in which case arcs from the root are only added implicitly (to any token that remains a root after parsing is completed). Empty stack initialization (which reduces the amount of nondeterminism in parsing) led to improved accuracy for Catalan, Chinese, Hungarian, Italian and Turkish.[4]

3. **Post-processing:** The baseline parser performs a single left-to-right pass over the input, but it is possible to allow a second pass where only unattached tokens are processed.[5] Such post-processing was found to improve results for Basque, Catalan, Czech, Greek and Hungarian.

Since the parsing algorithm only produces projective dependency graphs, we may use pseudo-projective parsing to recover non-projective dependencies, i.e., projectivize training data and encode information about these transformations in extended arc labels to support deprojectivization of the parser output (Nivre and Nilsson, 2005). Pseudo-projective parsing was found to have a positive effect on overall parsing accuracy only for Basque, Czech, Greek and Turkish. This result can probably be explained in terms of the frequency of non-projective dependencies in the different languages. For Basque, Czech, Greek and Turkish, more than 20% of the sentences have non-projective dependency graphs; for all the remaining languages the corresponding

[3] Complete specifications of all parameter settings for all languages, for both Single Malt and Blended, are available at http://w3.msi.vxu.se/users/jha/conll07/.

[4] For Arabic, Basque, Czech, and Greek, the lack of improvement can be explained by the fact that these data sets allow more than one label for dependencies from the artificial root. With empty stack initialization all such dependencies are assigned a default label, which leads to a drop in labeled attachment score. For English, however, empty stack initialization did not improve accuracy despite the fact that dependencies from the artificial root have a unique label.

[5] This technique is similar to the one used by Yamada and Matsumoto (2003), but with only a single post-processing pass parsing complexity remains linear in string length.

	Attributes					
Tokens	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	DEPREL
S: Top	+	+	+	+	+	+
S: Top−1				+		
I: Next	+	+	+	+	+	
I: Next+1	+			+		
I: Next+2				+		
I: Next+3				+		
G: Head of Top	+					
G: Leftmost dependent of Top						+
G: Rightmost dependent of Top						+
G: Leftmost dependent of Next						+

Figure 1: Baseline feature model (S = Stack, I = Input, G = Graph).

figure is 10% or less.[6]

The cumulative improvement after optimization of parsing algorithm parameters was a modest 0.32 percentage points on average over all ten languages, with a minimum of 0.00 (Arabic, English) and a maximum of 0.83 (Czech) (cf. table 1).

2.2 Feature Model

MaltParser uses a history-based feature model for predicting the next parsing action. Each feature of this model is an attribute of a token defined relative to the current stack S, input queue I, or partially built dependency graph G, where the attribute can be any of the symbolic input attributes in the CoNLL format: FORM, LEMMA, CPOSTAG, POSTAG and FEATS (split into atomic attributes), as well as the DEPREL attribute of tokens in the graph G. The baseline feature model is depicted in figure 1, where rows denote tokens, columns denote attributes, and each cell containing a plus sign represents a model feature.[7] This model is an extrapolation from many previous experiments on different languages and usually represents a good starting point for further optimization.

The baseline model was tuned for each of the ten languages using both forward and backward feature selection. The total number of features in the tuned models varies from 18 (Turkish) to 56 (Hungarian) but is typically between 20 and 30. This feature selection process constituted the major development effort for the Single Malt parser and also gave the greatest improvements in parsing accuracy, but since feature selection was to some extent interleaved with learning algorithm optimization, we only report the cumulative effect of both together in table 1.

2.3 Learning Algorithm

MaltParser supports several learning algorithms but the best results have so far been obtained with support vector machines, using the LIBSVM package (Chang and Lin, 2001). We use a quadratic kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$ and LIBSVM's built-in one-versus-one strategy for multi-class classification, converting symbolic features to numerical ones using the standard technique of binarization. As our baseline settings, we used $\gamma = 0.2$ and $r = 0$ for the kernel parameters, $C = 0.5$ for the penalty parameter, and $\epsilon = 1.0$ for the termination criterion. In order to reduce training times during development, we also split the training data for each language into smaller sets and trained separate multi-class classifiers for each set, using the POSTAG of Next as the defining feature for the split.

The time spent on optimizing learning algorithm parameters varies between languages, mainly due to lack of time. For Arabic, Basque, and Catalan, the baseline settings were used also in the dry run and final test. For Chinese, Greek and Hungarian,

[6] In fact, for Arabic, which has about 10% sentences with non-projective dependencies, it was later found that, with an optimized feature model, it is beneficial to projectivize the training data without trying to recover non-projective dependencies in the parser output. This was also the setting that was used for Arabic in the dry run and final test.

[7] The names Top and Next refer to the token on top of the stack S and the first token in the remaining input I, respectively.

	Development			Dry Run		Test		Test: UAS	
Language	**Base**	**PA**	**F+L**	**SM**	**B**	**SM**	**B**	**SM**	**B**
Arabic	70.31	70.31	71.67	70.93	73.09	74.75	76.52	84.21	85.81
Basque	73.86	74.44	76.99	77.18	80.12	74.97	76.92	80.61	82.84
Catalan	85.43	85.51	86.88	86.65	88.00	87.74	88.70	92.20	93.12
Chinese	83.85	84.39	87.64	87.61	88.61	83.51	84.67	87.60	88.70
Czech	75.00	75.83	77.74	77.91	82.17	77.22	77.98	82.35	83.59
English	85.44	85.44	86.35	86.35	88.74	85.81	88.11	86.77	88.93
Greek	72.67	73.04	74.42	74.89	78.17	74.21	74.65	80.66	81.22
Hungarian	74.62	74.64	77.40	77.81	80.04	78.09	80.27	81.71	83.55
Italian	81.42	81.64	82.50	83.37	85.16	82.48	84.40	86.26	87.77
Turkish	75.12	75.80	76.49	75.87	77.09	79.24	79.79	85.04	85.77
Average	77.78	78.10	79.81	79.86	82.12	79.80	81.20	84.74	86.13

Table 1: Development results for Single Malt (Base = baseline, PA = parsing algorithm, F+L = feature model and learning algorithm); dry run and test results for Single Malt (SM) and Blended (B) (with corrected test scores for Blended on Chinese). All scores are labeled attachment scores (LAS) except the last two columns, which report unlabeled attachment scores (UAS) on the test sets.

slightly better results were obtained by not splitting the training data into smaller sets; for the remaining languages, accuracy was improved by using the CPOSTAG of Next as the defining feature for the split (instead of POSTAG). With respect to the SVM parameters (γ, r, C, and ϵ), Arabic, Basque, Catalan, Greek and Hungarian retain the baseline settings, while the other languages have slightly different values for some parameters.

The cumulative improvement after optimization of feature model and learning algorithm parameters was 1.71 percentage points on average over all ten languages, with a minimum of 0.69 (Turkish) and a maximum of 3.25 (Chinese) (cf. table 1).

3 The Blended Parser

The Blended parser is an ensemble system based on the methodology proposed by Sagae and Lavie (2006). Given the output dependency graphs G_i ($1 \leq i \leq m$) of m different parsers for an input sentence x, we construct a new graph containing all the labeled dependency arcs proposed by some parser and weight each arc a by a score $s(a)$ reflecting its popularity among the m parsers. The output of the ensemble system for x is the maximum spanning tree of this graph (rooted at the node 0), which can be extracted using the Chu-Liu-Edmonds algorithm, as shown by McDonald et al. (2005). Following Sagae and Lavie (2006), we let $s(a) = \sum_{i=1}^{m} w_i^c a_i$, where w_i^c is the average labeled attachment score of parser i for the word class c[8] of the dependent of a, and a_i is 1 if $a \in G_i$ and 0 otherwise.

The Blended parser uses six component parsers, with three different parsing algorithms, each of which is used to construct one left-to-right parser and one right-to-left parser. The parsing algorithms used are the arc-eager baseline algorithm, the arc-standard variant of the baseline algorithm, and the incremental, non-projective parsing algorithm first described by Covington (2001) and recently used for deterministic classifier-based parsing by Nivre (2007), all of which are available in MaltParser. Thus, the six component parsers for each language were instances of the following:

1. Arc-eager projective left-to-right
2. Arc-eager projective right-to-left
3. Arc-standard projective left-to-right
4. Arc-standard projective right-to-left
5. Covington non-projective left-to-right
6. Covington non-projective right-to-left

[8] We use CPOSTAG to determine the part of speech.

	root		1		2		3–6		7+	
Parser	R	P	R	P	R	P	R	P	R	P
Single Malt	87.01	80.36	95.08	94.87	86.28	86.67	77.97	80.23	68.98	71.06
Blended	92.09	74.20	95.71	94.92	87.55	88.12	78.66	83.02	65.29	78.14

Table 2: Recall (R) and precision (P) of Single Malt and Blended for dependencies of different length, averaged over all languages (root = dependents of root node, regardless of length).

The final Blended parser was constructed by reusing the tuned Single Malt parser for each language (arc-standard left-to-right for Chinese, arc-eager left-to-right for the remaining languages) and training five additional parsers with the same parameter settings except for the following mechanical adjustments:

1. Pseudo-projective parsing was not used for the two non-projective parsers.

2. Feature models were adjusted with respect to the most obvious differences in parsing strategy (e.g., by deleting features that could never be informative for a given parser).

3. Learning algorithm parameters were adjusted to speed up training (e.g., by always splitting the training data into smaller sets).

Having trained all parsers on 90% of the training data for each language, the weights w_i^c for each parser i and coarse part of speech c was determined by the labeled attachment score on the remaining 10% of the data. This means that the results obtained in the dry run were bound to be overly optimistic for the Blended parser, since it was then evaluated on the same data set that was used to tune the weights.

Finally, we want to emphasize that the time for developing the Blended parser was severely limited, which means that several shortcuts had to be taken, such as optimizing learning algorithm parameters for speed rather than accuracy and using extrapolation, rather than proper tuning, for other important parameters. This probably means that the performance of the Blended system can be improved considerably by optimizing parameters for all six parsers separately.

4 Results and Discussion

Table 1 shows the labeled attachment score results from our internal dry run (training on 90% of the training data, testing on the remaining 10%) and the official test runs for both of our systems. It should be pointed out that the test score for the Blended parser on Chinese is different from the official one (75.82), which was much lower than expected due to a corrupted specification file required by MaltParser. Restoring this file and rerunning the parser on the Chinese test set, without retraining the parser or changing any parameter settings, resulted in the score reported here. This also improved the average score from 80.32 to 81.20, the former being the highest reported official score.

For the Single Malt parser, the test results are on average very close to the dry run results, indicating that models have not been overfitted (although there is considerably variation between languages). For the Blended parser, there is a drop of almost one percentage point, which can be explained by the fact that weights could not be tuned on held-out data for the dry run (as explained in section 3).

Comparing the results for different languages, we see a tendency that languages with rich morphology, usually accompanied by flexible word order, get lower scores. Thus, the labeled attachment score is below 80% for Arabic, Basque, Czech, Greek, Hungarian, and Turkish. By comparison, the more configurational languages (Catalan, Chinese, English, and Italian) all have scores above 80%. Linguistic properties thus seem to be more important than, for example, training set size, which can be seen by comparing the results for Italian, with one of the smallest training sets, and Czech, with one of the largest. The development of parsing methods that are better suited for morphologically rich languages with flexible word order appears as one of the most important goals for future research in this area.

Comparing the results of our two systems, we see that the Blended parser outperforms the Single Malt parser for all languages, with an average improvement of 1.40 percentage points, a minimum of 0.44 (Greek) and a maximum of 2.40 (English). As shown by McDonald and Nivre (2007), the Single Malt parser tends to suffer from two problems: error propagation due to the deterministic parsing strategy, typically affecting long dependencies more than short ones, and low precision on dependencies originating in the artificial root node due to fragmented parses.[9] The question is which of these problems is alleviated by the multiple views given by the component parsers in the Blended system. Table 2 throws some light on this by giving the precision and recall for dependencies of different length, treating dependents of the artificial root node as a special case. As expected, the Single Malt parser has lower precision than recall for root dependents, but the Blended parser has even lower precision (and somewhat better recall), indicating that the fragmentation is even more severe in this case.[10] By contrast, we see that precision and recall for other dependencies improve across the board, especially for longer dependencies, which probably means that the effect of error propagation is mitigated by the use of an ensemble system, even if each of the component parsers is deterministic in itself.

5 Conclusion

We have shown that deterministic, classifier-based dependency parsing, with careful optimization, can give highly accurate dependency parsing for a wide range of languages, as illustrated by the performance of the Single Malt parser. We have also demonstrated that an ensemble of deterministic, classifier-based dependency parsers, built on top of a tuned single-parser system, can give even higher accuracy, as shown by the results of the Blended parser, which has the highest labeled attachment score for five languages (Arabic, Basque, Catalan, Hungarian, and Italian), as well as the highest multilingual average score.

Acknowledgements

We want to thank all treebank providers for making the data available for the shared task and the (other) organizers for their efforts in organizing it. Special thanks to Ryan McDonald, for fruitful discussions and assistance with the error analysis, and to Kenji Sagae, for showing us how to produce a good blend. Thanks also to two reviewers for useful comments.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, pages 103–127.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (2003), chapter 13, pages 231–248.

M. A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proc. of the 39th Annual ACM Southeast Conf.*, pages 95–102.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[9] A fragmented parse is a dependency forest, rather than a tree, and is automatically converted to a tree by attaching all (other) roots to the artificial root node. Hence, children of the root node in the final output may not have been predicted as such by the treebank-induced classifier.

[10] This conclusion is further supported by the observation that the single most frequent "frame confusion" of the Blended parser, over all languages, is to attach two dependents with the label ROOT to the root node, instead of only one. The frequency of this error is more than twice as high for the Blended parser (180) as for the Single Malt parser (83).

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Human Language Technology Conf. and the Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL)*, pages 221–225.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

J. Nivre. 2007. Incremental non-projective dependency parsing. In *Human Language Technologies: The Annual Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 396–403.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proc. of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Probabilistic Parsing Action Models for Multi-lingual Dependency Parsing

Xiangyu Duan
Institute of Automation, Chinese Academy of Sciences
xyduan@nlpr.ia.ac.cn

Jun Zhao
Institute of Automation, Chinese Academy of Sciences
jzhao@nlpr.ia.ac.cn

Bo Xu
Institute of Automation, Chinese Academy of Sciences
xubo@hitic.ia.ac.cn

Abstract

Deterministic dependency parsers use parsing actions to construct dependencies. These parsers do not compute the probability of the whole dependency tree. They only determine parsing actions stepwisely by a trained classifier. To globally model parsing actions of all steps that are taken on the input sentence, we propose two kinds of probabilistic parsing action models that can compute the probability of the whole dependency tree. The tree with the maximal probability is outputted. The experiments are carried on 10 languages, and the results show that our probabilistic parsing action models outperform the original deterministic dependency parser.

1 Introduction

The target of CoNLL 2007 shared task (Nivre et al., 2007) is to parse texts in multiple languages by using a single dependency parser that has the capacity to learn from treebank data. Among parsers participating in CoNLL 2006 shared task (Buchholz et al., 2006), deterministic dependency parser shows great efficiency in time and comparable performances for multi-lingual dependency parsing (Nivre et al., 2006). Deterministic parser regards parsing as a sequence of parsing actions that are taken step by step on the input sentence. Parsing actions construct dependency relations between words.

Deterministic dependency parser does not score the entire dependency tree as most of state-of-the-art parsers. They only stepwisely choose the most probable parsing action. In this paper, to globally model parsing actions of all steps that are taken on the input sentence, we propose two kinds of probabilistic parsing action models that can compute the entire dependency tree's probability. Experiments are evaluated on diverse data set of 10 languages provided by CoNLL 2007 shared-task (Nivre et al., 2007). Results show that our probabilistic parsing action models outperform the original deterministic dependency parser. We also present a general error analysis across a wide set of languages plus a detailed error analysis of Chinese.

Next we briefly introduce the original deterministic dependency parsing algorithm that is a basic component of our models.

2 Introduction of Deterministic Dependency Parsing

There are mainly two representative deterministic dependency parsing algorithms proposed respectively by Nivre (2003), Yamada and Matsumoto (2003). Here we briefly introduce Yamada and Matsumoto's algorithm, which is adopted by our models, to illustrate deterministic dependency parsing. The other representative method of Nivre also parses sentences in a similar deterministic manner except different data structure and parsing actions.

Yamada's method originally focuses on unlabeled dependency parsing. Three kinds of parsing actions are applied to construct the dependency between two focus words. The two focus words are the current sub tree's root and the succeeding (right) sub tree's root given the current ***parsing state***. Every parsing step results in a ***new parsing state***, which includes all elements of the current partially built tree. Features are extracted about these two focus words. In the training phase, features and the corresponding parsing action compose the training

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 940–946,
Prague, June 2007. ©2007 Association for Computational Linguistics

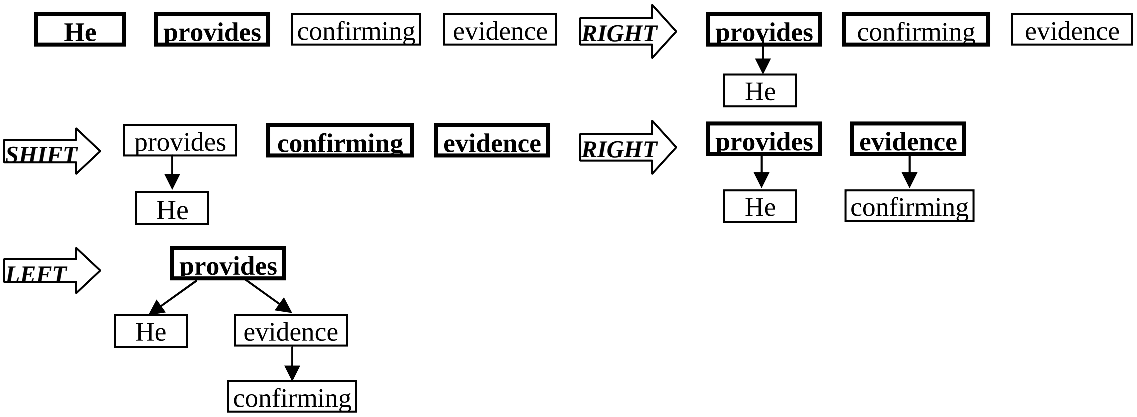


Figure 1. The example of the parsing process of Yamada and Matsumoto's method. The input sentence is "He provides confirming evidence."

data. In the testing phase, the classifier determines which parsing action should be taken based on the features. The parsing algorithm ends when there is no further dependency relation can be made on the whole sentence. The details of the three parsing actions are as follows:

LEFT: it constructs the dependency that the right focus word depends on the left focus word.

RIGHT: it constructs the dependency that the left focus word depends on the right focus word.

SHIFT: it does not construct dependency, just moves the parsing focus. That is, the new left focus word is the previous right focus word, whose succeeding sub tree's root is the new right focus word.

The illustration of these three actions and the parsing process is presented in figure 1. Note that the focus words are shown as bold black box.

We extend the set of parsing actions to do labeled dependency parsing. ***LEFT*** and ***RIGHT*** are concatenated by dependency labels, while ***SHIFT*** remains the same. For example in figure 1, the original action sequence "***RIGHT*** -> ***SHIFT*** -> ***RIGHT*** -> ***LEFT***" becomes "***RIGHT-SBJ*** -> ***SHIFT*** -> ***RIGHT-NMOD*** -> ***LEFT-OBJ***".

3 Probabilistic Parsing Action Models

Deterministic dependency parsing algorithms are greedy. They choose the most probable parsing action at every parsing step given the current parsing state, and do not score the entire dependency tree. To compute the probability of whole dependency tree, we propose two kinds of probabilistic models that are defined on parsing actins: parsing action chain model (PACM) and parsing action phrase model (PAPM).

3.1 Parsing Action Chain Model (PACM)

The parsing process can be viewed as a Markov Chain. At every parsing step, there are several candidate parsing actions. The objective of this model is to find the most probable sequence of parsing actions by taking the Markov assumption. As shown in figure 1, the action sequence "***RIGHT-SBJ*** -> ***SHIFT*** -> ***RIGHT-NMOD*** -> ***LEFT-OBJ***" constructs the right dependency tree of the example sentence. Choosing this action sequence among all candidate sequences is the objective of this model.

Firstly, we should define the probability of the dependency tree conditioned on the input sentence.

$$P(T \mid S) = \prod_{i=1...n} P(d_i \mid d_0 ... d_{i-1}, S) \qquad (1)$$

Where *T* denotes the dependency tree, *S* denotes the original input sentence, d_i denotes the parsing action at time step *i*. We add an artificial parsing action d_0 as initial action.

We introduce a variable $context_{d_i}$ to denote the resulting ***parsing state*** when the action d_i is taken on $context_{d_{i-1}}$. $context_{d_0}$ is the original input sentence.

Suppose $d_0 ... d_n$ are taken sequentially on the input sentence *S*, and result in a sequence of ***parsing states*** $context_{d_0} ... context_{d_n}$, then *P*(*T*|*S*) defined in equation (1) becomes as below:

$$\prod_{i=1...n} P(context_{d_i} \mid context_{d_0},...,context_{d_{i-1}}) \qquad (2)$$

$$\approx \prod_{i=1...n} P(context_{d_i} \mid context_{d_{i-1}}) \qquad (3)$$

$$= \prod_{i=1...n} P(d_i \mid context_{d_{i-1}}) \qquad (4)$$

Formula (3) comes from formula (2) by obeying the Markov assumption. Note that formula (4) is about the classifier of parsing actions. It denotes the probability of the parsing action d_i given the ***parsing state*** $context_{d_{i-1}}$. If we train a classifier that can predict with probability output, then we can compute *P(T|S)* by computing the product of the probabilities of parsing actions. The classifier we use throughout this paper is SVM (Vapnik, 1995). We adopt Libsvm (Chang and Lin, 2005), which can train multi-class classifier and support training and predicting with probability output (Chang and Lin, 2005).

For this model, the objective is to choose the parsing action sequence that constructs the dependency tree with the maximal probability.

$$\max P(T \mid S) = \max_{d_1...d_n} \prod_{i=1...n} P(d_i \mid context_{d_{i-1}}) \qquad (5)$$

Because this model chooses the most probable sequence, not the most probable parsing action at only one step, it avoids the greedy property of the original deterministic parsers.

We use beam search for the decoding of this model. We use *m* to denote the beam size. Then beam search is carried out as follows. At every parsing step, all ***parsing states*** are ordered (or partially *m* ordered) according to their probabilities. Probability of a ***parsing state*** is determined by multiplying the probabilities of actions that generate that state. Then we choose *m* best ***parsing states*** for this step, and next parsing step only consider these *m* best ***parsing states***. Parsing terminates when the first entire dependency tree is constructed. To obtain a list of *n*-best parses, we simply continue parsing until either *n* trees are found, or no further parsing can be fulfilled.

3.2 Parsing Action Phrase Model (PAPM)

In the Parsing Action Chain Model (PACM), actions are competing at every parsing step. Only *m* best parsing states resulted by the corresponding actions are kept for every step. But for the parsing problem, it is reasonable that actions are competing for which phrase should be built. For dependency syntax, one phrase consists of the head word and all its children. Based on this motivation, we propose Parsing Action Phrase Model (PAPM), which divides parsing actions into two classes: constructing action and shifting action.

If a phrase is built after an action is performed, the action is called constructing action. In original Yamada's algorithm, constructing actions are ***LEFT*** and ***RIGHT***. For example, if ***LEFT*** is taken, it indicates that the right focus word has found all its children and becomes the head of this new phrase. Note that one word with no children can also be viewed as a phrase if its dependency on other word is constructed. In the extended set of parsing actions for labeled parsing, compound actions, which consist of ***LEFT*** and ***RIGHT*** concatenated by dependency labels, are constructing actions.

If no phrase is built after an action is performed, the action is called shifting action. Such action is ***SHIFT***.

We denote a_j as constructing action and b_j as shifting action. *j* indexes the time step. Then we introduce a new concept: parsing action phrase. We use A_i to denote the *i*th parsing action phrase. It can be expanded as $A_i \rightarrow b_{j-k}...b_{j-1}a_j$. That is, parsing action phrase A_i is a sequence of parsing actions that constructs the next syntactic phrase.

For example, consider the parsing process in figure 1, A_1 is "***RIGHT-SBJ***", A_2 is "***SHIFT***, ***RIGHT-NMOD***", A_3 is "***LEFT-OBJ***". Note that A_1 consists of a constructing action, A_2 consists of a shifting action and a constructing action, A_3 consists of a constructing action.

The indexes are different for both sides of the expansion $A_i \rightarrow b_{j-k}...b_{j-1}a_j$, A_i is the *i*th parsing action phrase corresponding to both constructing action a_j at time step *j* and all its preceding shifting actions. Note that on the right side of the expansion, only one constructing action is allowed and is always at the last position, while shifting action can occur several times or does not occur at all. It is parsing action phrases, i.e. sequences of parsing actions, that are competing for which next phrase should be built.

The probability of the dependency tree given the input sentence is redefined as:

$$\begin{aligned}
P(T \mid S) &= \prod_{i=1 \ldots n} P(A_i \mid A_1 \ldots A_{i-1}, S) \qquad (6) \\
&= \prod_{i=1 \ldots n} P(context_{A_i} \mid context_{A_1} \ldots context_{A_{i-1}}) \\
&\approx \prod_{i=1 \ldots n} P(context_{A_i} \mid context_{A_{i-1}}) \\
&= \prod_{i=1 \ldots n} P(A_i \mid context_{A_{i-1}}) \\
&= \prod_{i=1 \ldots n} P(b_{j-k} \ldots b_{j-1} a_j \mid context_{A_{i-1}}) \\
&= \prod_{i=1 \ldots n} P(b_{j-k} \mid context_{A_{i-1}}) \cdot \\
&\quad (\prod_{t=2 \ldots k} P(b_{j-t+1} \mid context_{b_{j-t}})) \cdot P(a_j \mid context_{b_{j-1}})
\end{aligned}$$

Where k represents the number of steps that shifting action can be taken. $context_{A_i}$ is the parsing state resulting from a sequence of actions $b_{j-k} \ldots b_{j-1} a_j$ taken on $context_{A_{i-1}}$.

The objective in this model is to find the most probable sequence of parsing action phrases.

$$\max P(T \mid S) = \max_{A_1 \ldots A_n} \prod_{i=1 \ldots n} P(A_i \mid context_{A_{i-1}}) \qquad (7)$$

Similar with parsing action chain model (PACM), we use beam search for the decoding of parsing action phrase model (PAPM). The difference is that PAPM do not keep *m* best ***parsing states*** at every parsing step. Instead, PAPM keep *m* best states which are corresponding to *m* best current parsing action phrases (several steps of ***SHIFT*** and the last step of a constructing action).

4 Experiments and Results

Experiments are carried on 10 languages provided by CoNLL 2007 shared-task organizers (Nivre et al., 2007). Among these languages, Chinese (Chen et al., 2003), Catalan (Martí et al., 2007) and English (Johansson and Nugues, 2007) have low percentage of non-projective relations, which are 0.0%, 0.1% and 0.3% respectively. Except these three languages, we use software of projectivization/deprojectivization provided by Nivre and Nilsson (2005) for other languages. Because our algorithm only deals with projective parsing, we should projectivize training data at first to prepare for the following training of our algorithm. During testing, deprojectivization is applied to the output of the parser.

Considering the classifier of Libsvm (Chang and Lin, 2005), the features are extracted from the following fields of the data representation: FORM, LEMMA, CPOSTAG, POSTAG, FEATS and DEPREL. We split values of FEATS field into its atomic components. We only use available features of DEPREL field during deterministic parsing. We use similar feature context window as used in Yamada's algorithm (Yamada and Matsumoto, 2003). In detail, the size of feature context window is six, which consists of left two sub trees, two focus words related sub trees and right two sub trees. This feature template is used for all 10 languages.

4.1 Results of PACM and Yamada's Method

After submitting the testing results of Parsing Action Chain Model (PACM), we also perform original deterministic parsing proposed by Yamada and Matsumoto (2003). The total results are shown in table 1. The experimental results are mainly evaluated by labeled attachment score (*LAS*), unlabeled attachment score (*UAS*) and labeled accuracy (*LA*).

Table 1 shows that Parsing Action Chain Model (PACM) outperform original Yamada's parsing method for all languages. The *LAS* improvements range from 0.60 percentage points to 1.71 percentage points. Note that the original Yamada's method still gives testing results above the official reported average performance of all languages.

	Ara	**Bas**	**Cat**	**Chi**	**Cze**	**Eng**	**Gre**	**Hun**	**Ita**	**Tur**
LAS_{Yam}	69.31	69.67	83.26	81.88	74.63	84.81	72.75	76.24	80.08	73.94
UAS_{Yam}	78.93	75.86	88.53	86.17	80.11	85.83	79.45	79.97	83.69	79.79
LA_{Yam}	81.13	75.71	88.36	84.56	82.10	89.71	82.58	88.37	86.93	80.81
LAS_{PACM}	**69.91**	**71.26**	**84.95**	**82.58**	**75.34**	**85.83**	**74.29**	**77.06**	**80.75**	**75.03**
UAS_{PACM}	**79.04**	**77.57**	**89.71**	**86.88**	**80.82**	**86.97**	**80.77**	**80.66**	**84.20**	**81.03**
LA_{PACM}	**81.40**	**77.35**	**89.55**	**85.35**	**83.17**	**90.57**	**83.87**	**88.92**	**87.32**	**81.17**

Table 1. The performances of Yamada's method (*Yam*) and Parsing Action Chain Model (*PACM*).

4.2 Results of PAPM

Not all languages have only one root node of a sentence. Since Parsing Action Phrase Model (PAPM) only builds dependencies, and shifting action is not the ending action of a parsing action phrase, PAPM always ends with one root word. This property makes PAPM only suitable for Catalan, Chinese, English and Hungarian, which are unary root languages. PAPM result of Catalan was not submitted before deadline due to the shortage of time and computing resources. We report Catalan's PAPM result together with that of other three languages in table 2.

	Cat	Chi	Eng	Hun
LAS_{PAPM}	87.26	82.64	86.69	76.89
UAS_{PAPM}	92.07	86.94	87.87	80.53
LA_{PAPM}	91.89	85.41	92.04	89.73

Table 2. The performance of Parsing Action Phrase Model (*PAPM*) for Catalan, Chinese, English and Hungarian.

Compared with the results of PACM shown in table 1, the performance of PAPM differs among different languages. Catalan and English show that PAPM improves 2.31% and 0.86% respectively over PACM, while the improvement of Chinese is marginal, and there is a little decrease of Hungarian. Hungarian has relatively high percentage of non-projective relations. If phrase consists of head word and its non-projective children, the constructing actions that are main actions in PAPM will be very difficult to be learned because some non-projective children together with their heads have no chance to be simultaneously as focus words. Although projectivization is also performed for Hungarian, the built-in non-projective property still has negative influence on the performance.

5 Error Analysis

In the following we provide a general error analysis across a wide set of languages plus a detailed analysis of Chinese.

5.1 General Error Analysis

One of the main difficulties in dependency parsing is the determination of long distance dependencies. Although all kinds of evaluation scores differ dramatically among different languages, 69.91% to 85.83% regarding *LAS*, there are some general observations reflecting the difficulty of long distance dependency parsing. We study this difficulty from two aspects about our full submission of PACM: precision of dependencies of different arc lengths and precision of root nodes.

For arcs of length 1, all languages give high performances with lowest 91.62% of Czech (Böhmova et al., 2003) to highest 96.8% of Catalan (Martí et al., 2007). As arcs lengths grow longer, various degradations are caused. For Catalan, score of arc length 2 is similar with that of arc length 1, but there are dramatic degradations for longer arc lengths, from 94.94% of arc length 2 to 85.22% of length 3-6. For English (Johansson and Nugues, 2007) and Italian (Montemagni et al., 2003), there are graceful degradation for arcs of length 1,2 and 3-6, with 96-91-85 of English and 95-85-75 of Italian. For other languages, long arcs also give remarkable degradations that pull down the performance.

Precision of root nodes also reflects the performance of long arc dependencies because the arc between the root and its children are often long arcs. In fact, it is the precision of roots and arcs longer than 7 that mainly pull down the overall performance. Yamada's method is a bottom-up parsing algorithm that builds short distance dependencies at first. The difficulty of building long arc dependencies may partially be resulted from the errors of short distance dependencies. The deterministic manner causes error propagation, and it indirectly indicates that the errors of roots are the final results of error propagation of short distance dependencies. But there is an exception occurred in Chinese. The root precision is 90.48%, only below the precision of arcs of length 1. This phenomenon exists because the sentences in Chinese data set (Chen et al., 2003) are in fact clauses with average length of 5.9 rather than entire sentences. The root words are heads of clauses.

Both Parsing Action Chain Model (PACM) and Parsing Action Phrase Model (PAPM) avoid greedy property of original Yamada's method. It can be expected that there will be a precision improvement of long distance dependencies over original Yamada's method. For PACM, the results of Basque (Aduriz et al., 2003), Catalan (Martí et al., 2007), Chinese (Chen et al., 2003), English (Johansson and Nugues, 2007) and Greek (Prokopidis et al., 2005) show that the root precision improvement over Yamada's method is more conspicuous than that of other long distance dependencies. The largest improvement of roots precision is 10.7% of Greek. While for Arabic (Hajic et al., 2004), Czech (Böhmova et al., 2003), Hungarian (Csendes et al., 2005), Italian (Montemagni et al., 2003) and Turkish (Oflazer et al., 2003), the improvement of root precision is small, but dependencies of arcs longer than 1 give better scores. For PAPM, good performances of Catalan and English also give significant improvements of root precision over PACM. For Catalan, the root precision improvement is from 63.86% to 95.21%; for English, the root precision improvement is from 62.03% to 89.25%.

5.2 Error Analysis of Chinese

There are mainly two sources of errors regarding *LAS* in Chinese dependency parsing.

One is from conjunction words (C) that have a relatively high percentage of wrong heads (about 20%), and therefore 19% wrong dependency labels. In Chinese, conjunction words often concatenate clauses. Long distance dependencies between clauses are bridged by conjunction words. It is difficult for conjunction words to find their heads.

The other source of errors comes from auxiliary words (DE) and preposition words (P). Unlike conjunction words, auxiliary words and preposition words have high performance of finding right head, but label accuracy (LA) decrease significantly. The reason may lie in the large dependency label set consisting of 57 kinds of dependency labels in Chinese. Moreover, auxiliary words (DE) and preposition words (P) have more possible dependency labels than other coarse POS have. This introduces ambiguity for parsers.

Most common POS including noun and verb contribute much to the overall performance of 83% Labeled Attachment Scores (*LAS*). Adverbs obtain top score while adjectives give the worst.

6 Conclusion

We propose two kinds of probabilistic models defined on parsing actions to compute the probability of entire sentence. Compared with original Yamada and Matsumoto's deterministic dependency method which stepwisely chooses most probable parsing action, the two probabilistic models improve the performance regarding all 10 languages in CoNLL 2007 shared task. Through the study of parsing results, we find that long distance dependencies are hard to be determined for all 10 languages. Further analysis about this difficulty is needed to guide the research direction. Feature exploration is also necessary to provide more informative features for hard problems.

Ackowledgements

This work was supported by Hi-tech Research and Development Program of China under grant No. 2006AA01Z144, the Natural Sciences Foundation of China under grant No. 60673042, and the Natural Science Foundation of Beijing under grant No. 4052027, 4073043.

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. SIGNLL.

Chih-Chung Chang and Chih-Jen Lin. 2005. LIBSVM: A library for support vector machines.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT).*

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL-2005*, pages 99–106.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, S. Marinov. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proc. of the Tenth Conference on Computational Natural Language Learning (CoNLL).*

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT).*

V. Vapnik. 1995. The Nature of StatisticalLearning Theory. Springer.

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia and M. Oronoz. 2003. Construction of a Basque Dependency Treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajic, E. Hajicová and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, 103–127.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang and Z. Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé (2003), chapter 13, pages 231–248.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajic, O. Smrz, P. Zemánek, J. Snaidauf and E. Beska. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez and M. Bertran. 2007. CESS-ECE: A Multilingual and Multilevel Annotated Corpus. Available for download from: http://www.lsi.upc.edu/~mbertran/cess-ece/.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek depen- dency treebank. In Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT), pages 149–160.

Fast and Robust Multilingual Dependency Parsing with a Generative Latent Variable Model

Ivan Titov
University of Geneva
24, rue Général Dufour
CH-1211 Genève 4, Switzerland
ivan.titov@cui.unige.ch

James Henderson
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, United Kingdom
james.henderson@ed.ac.uk

Abstract

We use a generative history-based model to predict the most likely derivation of a dependency parse. Our probabilistic model is based on Incremental Sigmoid Belief Networks, a recently proposed class of latent variable models for structure prediction. Their ability to automatically induce features results in multilingual parsing which is robust enough to achieve accuracy well above the average for each individual language in the multilingual track of the CoNLL-2007 shared task. This robustness led to the third best overall average labeled attachment score in the task, despite using no discriminative methods. We also demonstrate that the parser is quite fast, and can provide even faster parsing times without much loss of accuracy.

1 Introduction

The multilingual track of the CoNLL-2007 shared task (Nivre et al., 2007) considers dependency parsing of texts written in different languages. It requires use of a single dependency parsing model for the entire set of languages; model parameters are estimated individually for each language on the basis of provided training sets. We use a recently proposed dependency parser (Titov and Henderson, 2007b)[1] which has demonstrated state-of-the-art performance on a selection of languages from the CoNLL-X shared task (Buchholz and Marsi, 2006). This parser employs a latent variable model, Incremental Sigmoid Belief Networks (ISBNs), to define a generative history-based model of projective parsing. We used the pseudo-projective transformation introduced in (Nivre and Nilsson, 2005) to cast non-projective parsing tasks as projective. Following (Nivre et al., 2006), the encoding scheme called *HEAD* in (Nivre and Nilsson, 2005) was used to encode the original non-projective dependencies in the labels of the projectivized dependency tree. In the following sections we will briefly discuss our modifications to the ISBN parser, experimental setup, and achieved results.

2 The Probability Model

Our probability model uses the parsing order proposed in (Nivre et al., 2004), but instead of performing deterministic parsing as in (Nivre et al., 2004), this ordering is used to define a generative history-based model, by adding word prediction to the *Shift* parser action. We also decomposed some parser actions into sub-sequences of decisions. We split arc prediction decisions (*Left-Arc*$_r$ and *Right-Arc*$_r$) each into two elementary decisions: first the parser creates the corresponding arc, then it assigns a relation r to the arc. Similarly, we decompose the decision to shift a word into a decision to shift and a prediction of the word. We used part-of-speech tags and fine-grain word features, which are given in the data, to further decompose word predictions. First we predict the fine-grain part-of-speech tag for the word, then the set of word features (treating each set as an atomic value), and only then the particu-

[1] The ISBN parser will be soon made downloadable from the authors' web-page.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 947–951,
Prague, June 2007. ©2007 Association for Computational Linguistics

lar word form. This approach allows us to both decrease the effect of sparsity and to avoid normalization across all the words in the vocabulary, significantly reducing the computational expense of word prediction. When conditioning on words, we treated each word feature individually, as this proved to be useful in (Titov and Henderson, 2007b).

The probability of each parser decision, conditioned on the complete parse history, is modeled using a form a graphical model called Incremental Sigmoid Belief Networks. ISBNs, originally proposed for constituent parsing in (Titov and Henderson, 2007a), use vectors of binary latent variables to encode information about the parse history. These history variables are similar to the hidden state of a Hidden Markov Model. But unlike the graphical model for an HMM, which would specify conditional dependency edges only between adjacent states in the parse history, the ISBN graphical model can specify conditional dependency edges between latent variables which are arbitrarily far apart in the parse history. The source state of such an edge is determined by the partial parse structure built at the time of the destination state, thereby allowing the conditional dependency edges to be appropriate for the structural nature of the parsing problem. In particular, they allow conditional dependencies to be local in the parse structure, not just local in the history sequence. In this they are similar to the class of neural networks proposed in (Henderson, 2003) for constituent parsing. In fact, in (Titov and Henderson, 2007a) it was shown that this neural network can be viewed as a coarse approximation to the corresponding ISBN model.

Traditional statistical parsing models also condition on features which are local in the parse structure, but these features need to be explicitly defined before learning, and require careful feature selection. This is especially difficult for languages unknown to the parser developer, since the number of possible features grows exponentially with the structural distance considered.

The ISBN model uses an alternative approach, where latent variables are used to induce features during learning. The most important problem in designing an ISBN is to define an appropriate structural locality for each parser decision. This is done by choosing a fixed set of relationships between parser states, where the information which is needed to make the decision at the earlier state is also useful in making the decision at the later state. The latent variables for these related states are then connected with conditional dependency edges in the ISBN graphical model. Longer conditional dependencies are then possible through chains of these immediate conditional dependencies, but there is an inductive bias toward shorter chains. This bias makes it important that the set of chosen relationships defines an appropriate notion of locality. However, as long as there exists some chain of relationships between any two states, then any statistical dependency which is clearly manifested in the data can be learned, even if it was not foreseen by the designer. This provides a potentially powerful form of feature induction, which is nonetheless biased toward a notion of locality appropriate for the nature of the problem.

In our experiments we use the same definition of structural locality as was proposed for the ISBN dependency parser in (Titov and Henderson, 2007b). The current state is connected to previous states using a set of 7 distinct relationships defined in terms of each state's parser configuration, which includes of a stack and a queue. Specifically, the current state is related to the last previous state whose parser configuration has: the same queue, the same stack, a stack top which is the rightmost right child of the current stack top, a stack top which is the leftmost left child of the current stack top, a front of the queue which is the leftmost child of the front of the current queue, a stack top which is the head word of the current stack top, a front of the queue which is the current stack top. Different model parameters are trained for each of these 7 types of relationship, but the same parameters are used everywhere in the graphical model where the relationship holds.

Each latent variable in the ISBN parser is also conditionally dependent on a set of explicit features of the parsing history. As long as these explicit features include all the new information from the last parser decision, the performance of the model is not very sensitive to this design choice. We used the base feature model defined in (Nivre et al., 2006) for all the languages but Arabic, Chinese, Czech, and Turkish. For Arabic, Chinese, and Czech, we used the same feature models used in the CoNLL-X shared task by (Nivre et al., 2006), and for Turkish we used again the base feature model but extended it with a single feature: the part-of-speech tag of the token preceding the current top of the stack.

3 Parsing

Exact inference in ISBN models is not tractable, but effective approximations were proposed in (Titov and Henderson, 2007a). Unlike (Titov and Henderson, 2007b), in the shared task we used only the simplest feed-forward approximation, which replicates the computation of a neural network of the type proposed in (Henderson, 2003). We would expect better performance with the more accurate approximation based on variational inference proposed and evaluated in (Titov and Henderson, 2007a). We did not try this because, on larger treebanks it would have taken too long to tune the model with this better approximation, and using different approximation methods for different languages would not be compatible with the shared task rules.

To search for the most probable parse, we use the heuristic search algorithm described in (Titov and Henderson, 2007b), which is a form of beam search. In section 4 we show that this search leads to quite efficient parsing.

To overcome a minor shortcoming of the parsing algorithm of (Nivre et al., 2004) we introduce a simple language independent post-processing step. Nivre's parsing algorithm allows unattached nodes to stay on the stack at the end of parsing, which is reasonable for treebanks with unlabeled attachment to root. However, this sometimes happens with languages where only labeled attachment to root is allowed. In these cases (only 35 tokens in Greek, 17 in Czech, 1 in Arabic, on the final testing set) we attached them using a simple rule: if there are no tokens in the sentence attached to root, then the considered token is attached to root with the most frequent root-attachment relation used for its part-of-speech tag. If there are other root-attached tokens in the sentence, it is attached to the next root-attached token with the most frequent relation. Preference is given to the most frequent attachment direction for its part-of-speech tag. This rule guarantees that no loops are introduced by the post-processing.

4 Experiments

We evaluated the ISBN parser on all the languages considered in the shared task (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003). ISBN models were trained using a small development set taken out from the training set, which was used for tuning learning and decoding parameters, for early stopping and very coarse feature engineering.[2] The sizes of the development sets were different: starting from less than 2,000 tokens for smaller treebanks to 5,000 tokens for the largest one. The relatively small sizes of the development sets limited our ability to perform careful feature selection, but this should not have significantly affected the model performance, as discussed in section 2.[3] We used frequency cutoffs: we ignored any property (word form, lemma, feature) which occurs in the training set less than a given threshold. We used a threshold of 20 for Greek and Chinese and a threshold of 5 for the rest. Because cardinalities of each of these sets (sets of word forms, lemmas and features) effect the model efficiency, we selected the larger threshold when validation results with the smaller threshold were comparable. For the ISBN latent variables, we used vectors of length 80, based on our previous experience.

Results on the final testing set are presented in table 1. The model achieves relatively high scores on each individual language, significantly better than each average result in the shared task. This leads to the third best overall average results in the shared task, both in average labeled attachment score and in average unlabeled attachment score. The absolute error increase in labeled attachment score over the best system is only 0.4%. We attribute ISBN's success mainly to its ability to automatically induce features, as this significantly reduces the risk of omitting any important highly predictive features. This makes an ISBN parser a particularly good baseline when considering a new treebank or language, be-

[2] We plan to make all the learning and decoding parameters available on our web-page.

[3] Use of cross-validation with our model is relatively time-consuming and, thus, not quite feasible for the shared task.

	Ara	Bas	Cat	Chi	Cze	Eng	Gre	Hun	Ita	Tur	**Ave**
LAS	74.1	75.5	87.4	82.1	77.9	88.4	73.5	77.9	82.3	79.8	**79.90**
UAS	83.2	81.9	93.4	87.9	84.2	89.7	81.2	82.2	86.3	86.2	**85.62**

Table 1: Labeled attachment score (LAS) and unlabeled attachment score (UAS) on the final testing sets

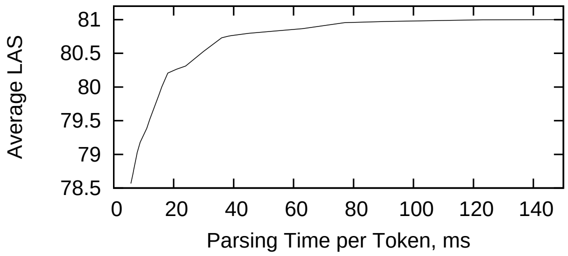


Figure 1: Average labeled attachment score on Basque, Chinese, English, and Turkish development sets as a function of parsing time per token

cause it does not require much effort in feature engineering. As was demonstrated in (Titov and Henderson, 2007b), even a minimal set of local explicit features achieves results which are non-significantly different from a carefully chosen set of explicit features, given the language independent definition of locality described in section 2.

It is also important to note that the model is quite efficient. Figure 1 shows the tradeoff between accuracy and parsing time as the width of the search beam is varied, on the development set. This curve plots the average labeled attachment score over Basque, Chinese, English, and Turkish as a function of parsing time per token.[4] Accuracy of only 1% below the maximum can be achieved with average processing time of 17 ms per token, or 60 tokens per second.[5]

We also refer the reader to (Titov and Henderson, 2007b) for more detailed analysis of the ISBN dependency parser results, where, among other things, it was shown that the ISBN model is especially accurate at modeling long dependencies.

5 Conclusion

We evaluated the ISBN dependency parser in the multilingual shared task setup and achieved competitive accuracy on every language, and the third best average score overall. The proposed model requires minimal design effort because it relies mostly on automatic feature induction, which is highly desirable when using new treebanks or languages. The parsing time needed to achieve high accuracy is also quite small, making this model a good candidate for use in practical applications.

The fact that our model defines a probability model over parse trees, unlike the previous state-of-the-art methods (Nivre et al., 2006; McDonald et al., 2006), makes it easier to use this model in applications which require probability estimates, such as in language processing pipelines or for language modeling. Also, as with any generative model, it should be easy to improve the parser's accuracy with discriminative reranking, such as discriminative retraining techniques (Henderson, 2004) or data-defined kernels (Henderson and Titov, 2005), with or even without the introduction of any additional linguistic features.

Acknowledgments

This work was funded by Swiss NSF grant 200020-109685, UK EPSRC grant EP/E019501/1, and EU FP6 grant 507802 for project TALK.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

[4] A piecewise-linear approximation for each individual language was used to compute the average. Experiments were run on a standard 2.4 GHz desktop PC.

[5] For Basque, Chinese, and Turkish this time is below 7 ms, but for English it is 38 ms. English, along with Catalan, required the largest beam across all 10 languages. Note that accuracy in the lowest part of the curve can probably be improved by varying latent vector size and frequency cut-offs. Also, efficiency was not the main goal during the implementation of the parser, and it is likely that a much faster implementation is possible.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the Tenth Conference on Computational Natural Language Learning*, New York, USA.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank.* Springer.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

James Henderson and Ivan Titov. 2005. Data-defined kernels for parse reranking derived from probabilistic models. In *Proc. 43rd Meeting of Association for Computational Linguistics*, Ann Arbor, MI.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. joint meeting of North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conf.*, pages 103–110, Edmonton, Canada.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proc. 42nd Meeting of Association for Computational Linguistics*, Barcelona, Spain.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of the Tenth Conference on Computational Natural Language Learning*, New York, USA.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. 43rd Meeting of Association for Computational Linguistics*, pages 99–106, Ann Arbor, MI.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proc. of the Eighth Conference on Computational Natural Language Learning*, pages 49–56, Boston, USA.

Joakim Nivre, Johan Hall, Jens Nilsson, Gulsen Eryigit, and Svetoslav Marinov. 2006. Pseudo-projective dependency parsing with support vector machines. In *Proc. of the Tenth Conference on Computational Natural Language Learning*, pages 221–225, New York, USA.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

K. Oflazer, B. Say, D. Zeynep Hakkani-T ür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

Ivan Titov and James Henderson. 2007a. Constituent parsing with incremental sigmoid belief networks. In *Proc. 45th Meeting of Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Ivan Titov and James Henderson. 2007b. A latent variable model for generative dependency parsing. In *Proc. 10th Int. Conference on Parsing Technologies (IWPT)*, Prague, Czech Republic.

Multilingual Dependency Parsing using Global Features

Tetsuji Nakagawa
Oki Electric Industry Co., Ltd.
2−5−7 Honmachi, Chuo-ku, Osaka 541−0053, Japan
nakagawa378@oki.com

Abstract

In this paper, we describe a two-stage multilingual dependency parser used for the multilingual track of the CoNLL 2007 shared task. The system consists of two components: an unlabeled dependency parser using Gibbs sampling which can incorporate sentence-level (global) features as well as token-level (local) features, and a dependency relation labeling module based on Support Vector Machines. Experimental results show that the global features are useful in all the languages.

1 Introduction

Making use of as many informative features as possible is crucial to obtain high performance in machine learning based NLP. Recently, several methods for incorporating non-local features have been investigated, though such features often make models complex and thus complicate inference. Collins and Koo (2005) proposed a reranking method for phrase structure parsing with which any type of global features in a parse tree can be used. For dependency parsing, McDonald and Pereira (2006) proposed a method which can incorporate some types of global features, and Riedel and Clarke (2006) studied a method using integer linear programming which can incorporate global linguistic constraints. In this paper, we study dependency parsing using Gibbs sampling which can incorporate any type of global feature in a sentence. The parser determines unlabeled dependency structures only, and we attach dependency relation labels using Support Vector Machines afterwards.

We participated in the multilingual track of the CoNLL 2007 shared task (Nivre et al., 2007), and evaluated the system on data sets of 10 languages (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003).

The rest of the paper describes the specification of the system and the evaluation results.

2 Unlabeled Dependency Parsing using Global Features

2.1 Probabilistic Model

Rosenfeld et al. (2001) proposed whole-sentence exponential language models which can incorporate arbitrary features in a sentence, and we consider here a similar probabilistic model for dependency parsing which can incorporate any sentence-level feature. Let $\mathbf{w} = w_1 \cdots w_{|\mathbf{w}|}$ denote an input sentence consisting of $|\mathbf{w}|$ tokens, and $\mathbf{h} = h_1 \cdots h_{|\mathbf{w}|}$ denote the sequence of the indices of each token's head. Root nodes of a sentence do not have heads, and we regard the index of a root node's head as zero, i.e., $h_i \in \{0, 1, \cdots, |\mathbf{w}|\} \setminus \{i\}$. We define the probability distribution of the dependency structure $\mathbf{h}$ given a sentence $\mathbf{w}$ using exponential models as follows:

$$P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w}) = \frac{1}{Z_{\Lambda,\mathrm{M}}(\mathbf{w})} Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w}) \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(\mathbf{w},\mathbf{h})\right\}, \quad (1)$$

$$Z_{\Lambda,\mathrm{M}}(\mathbf{w}) = \sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{w})} Q_{\mathrm{M}}(\mathbf{h}'|\mathbf{w}) \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(\mathbf{w},\mathbf{h}')\right\}, \quad (2)$$

where $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ is an initial distribution, $f_k(\mathbf{w},\mathbf{h})$ is the k-th feature function, K is the number of feature functions, and λ_k is the weight of the k-th feature. $\mathcal{H}(\mathbf{w})$ is the set of possible configurations of heads for a given sentence $\mathbf{w}$. Although it is appropriate that $\mathcal{H}(\mathbf{w})$ is the set of projective trees for projective languages, and is the set of non-projective trees (which is a superset of the set of projective trees) for non-projective languages, in this study, we define $\mathcal{H}(\mathbf{w})$ to be the set of all the possible graphs, which contains $|\mathbf{w}|^{|\mathbf{w}|}$ elements. $P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w})$ and $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ are defined over $\mathcal{H}(\mathbf{w})$[1]. The probability distribution $P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w})$ is a joint distribution of all the heads conditioned by a sentence, therefore we call this model *sentence-level model*. The feature function $f_k(\mathbf{w},\mathbf{h})$ is defined on a sentence $\mathbf{w}$ with heads $\mathbf{h}$, and we can use any information in the sentence without the independence assumption for the heads of the tokens, therefore we call $f_k(\mathbf{w},\mathbf{h})$

[1] $\mathcal{H}(\mathbf{w})$ is a superset of the set of non-projective trees, and is an unnecessarily large set which contains ill-formed dependency trees such as trees with cycles. This issue may cause reduction of parsing performance, but we adopt this approach for computational efficiency.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 952–956,
Prague, June 2007.

sentence-level (global) feature. We define initial distribution $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ as the product of $q_{\mathrm{M}}(h|\mathbf{w},t)$ which is the probability distribution of the head h of each t-th token calculated with maximum entropy models:

$$Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w}){=}\prod_{t=1}^{|\mathbf{w}|} q_{\mathrm{M}}(h_t|\mathbf{w},t), \qquad (3)$$

$$q_{\mathrm{M}}(h|\mathbf{w},t){=}\frac{1}{Y_{\mathrm{M}}(\mathbf{w},t)}\exp\left\{\sum_{l=1}^{L}\mu_l g_l(\mathbf{w},t,h)\right\}, \qquad (4)$$

$$Y_{\mathrm{M}}(\mathbf{w},t){=}\sum_{\substack{h'=0\\h'\neq t}}^{|\mathbf{w}|}\exp\left\{\sum_{l=1}^{L}\mu_l g_l(\mathbf{w},t,h')\right\}, \qquad (5)$$

where $g_l(\mathbf{w},t,h)$ is the l-th feature function, L is the number of feature functions, and μ_l is the weight of the l-th feature. $q_{\mathrm{M}}(h|\mathbf{w},t)$ is a model of the head of a single token, calculated independently from other tokens, therefore we call $q_{\mathrm{M}}(h|\mathbf{w},t)$ *token-level model*, and $g_l(\mathbf{w},t,h)$ *token-level (local) feature*.

2.2 Decoding and Parameter Estimation

Let us consider how to find the optimal solution $\hat{\mathbf{h}}$, given a sentence $\mathbf{w}$, parameters of the sentence-level model $\Lambda = \{\lambda_1,\cdots,\lambda_K\}$, and parameters of the token-level model $\mathrm{M} = \{\mu_1,\cdots,\mu_L\}$. Since the probabilistic model contains global features and efficient algorithms such as dynamic programming cannot be used, we use Gibbs sampling to obtain an approximated solution. Gibbs sampling can efficiently generate samples from high-dimensional probability distributions with complex dependencies among variables (Andrieu et al., 2003), and we assume that R samples $\{\mathbf{h}^{(1)},\cdots,\mathbf{h}^{(R)}\}$ are generated from $P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w})$ using Gibbs sampling. Then, the marginal distribution of the head of the t-th token given $\mathbf{w}$, $P_t(h|\mathbf{w})$, is approximately calculated as follows:

$$\begin{aligned}P_t(h|\mathbf{w}) &= \sum_{\substack{h_1,\cdots,h_{t-1},h_{t+1},\cdots,h_{|\mathbf{w}|}\\h_t=h}} P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w}),\\ &{=}\sum_{\mathbf{h}} P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w})\delta(h,h_t) \simeq \frac{1}{R}\sum_{r=1}^{R}\delta(h,h_t^{(r)}), \qquad (6)\end{aligned}$$

where $\delta(i,j)$ is the Kronecker delta. In order to find a solution using the marginal distribution, we adopt the maximum spanning tree (MST) framework proposed by McDonald et al. (2005a). In this framework, scores for possible edges in dependency graphs are defined, and the optimal dependency tree is found as the MST in which the summation of the edge scores is maximized. Let $s(i,j)$ denote the score of the edge from a parent node (head) i to a child node (dependent) j. We define $s(i,j)$ as follows:

$$s(i,j){=}\log P_j(i|\mathbf{w}). \qquad (7)$$

We use the logarithm of the marginal distribution because the summation of edge scores is maximized by the MST search algorithms but the product of the marginal distributions should be maximized. The best projective parse tree is obtained using the Eisner algorithm (Eisner, 1996) with the scores, and the best non-projective one is obtained using the Chu-Liu-Edmonds (CLE) algorithm (McDonald et al., 2005b).

Although in this method, the factored score $s(i,j)$ is used to measure likelihood of dependency trees, the score is calculated taking a whole sentence into consideration using Gibbs sampling.

Next, we explain how to estimate the parameters of our models, given training data consisting of N examples $\{\langle\mathbf{w}^1,\mathbf{h}^1\rangle,\cdots,\langle\mathbf{w}^N,\mathbf{h}^N\rangle\}$. In order to estimate the parameters of the token-level model $\mathrm{M} = \{\mu_1,\cdots,\mu_L\}$, we use maximum a posteriori estimation with Gaussian priors. We define the following objective function $\mathcal{M}$:

$$\mathcal{M}{=}\log\prod_{n=1}^{N} Q_{\mathrm{M}}(\mathbf{h}^n|\mathbf{w}^n) - \frac{1}{2\sigma^2}\sum_{l=1}^{L}\mu_l^2, \qquad (8)$$

where σ is a hyper parameter of Gaussian priors. The optimal parameters M which maximize $\mathcal{M}$ can be obtained by quasi-Newton methods such as the L-BFGS algorithm with above $\mathcal{M}$ and its partial derivatives. The parameters of the sentence-level model $\Lambda = \{\lambda_1,\cdots,\lambda_K\}$ can also be estimated in a similar way with the following objective function $\mathcal{L}$ after the parameters of the token-level model are estimated.

$$\mathcal{L}{=}\log\prod_{n=1}^{N} P_{\Lambda,\mathrm{M}}(\mathbf{h}^n|\mathbf{w}^n) - \frac{1}{2\sigma'^2}\sum_{k=1}^{K}\lambda_k^2. \qquad (9)$$

This objective function and its partial derivative contain summations over all the possible configurations which are difficult to calculate. We approximately calculate these values using static Monte Carlo (not MCMC) methods with fixed S samples $\{\mathbf{h}^{n(1)},\cdots,\mathbf{h}^{n(S)}\}$ generated from $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w}^n)$[2]:

$$\log Z_{\Lambda,\mathrm{M}}(\mathbf{w}^n){\simeq}\log\frac{1}{S}\sum_{s=1}^{S}\exp\left\{\sum_{k=1}^{K}\lambda_k f_k(\mathbf{w}^n,\mathbf{h}^{n(s)})\right\}, (10)$$

$$\begin{aligned}&\sum_{\mathbf{h}'\in\mathcal{H}(\mathbf{w}^n)} P_{\Lambda,\mathrm{M}}(\mathbf{h}'|\mathbf{w}^n) f_k(\mathbf{w}^n,\mathbf{h}')\\ &\simeq\frac{1}{S}\sum_{s=1}^{S}\frac{f_k(\mathbf{w}^n,\mathbf{h}^{n(s)})}{Z_{\Lambda,\mathrm{M}}(\mathbf{w}^n)}\exp\left\{\sum_{k'=1}^{K}\lambda_{k'} f_{k'}(\mathbf{w}^n,\mathbf{h}^{n(s)})\right\}. \qquad (11)\end{aligned}$$

[2] Static Monte Carlo methods become inefficient when the dimension of the probabilistic distribution is high, and more sophisticated methods would be used for accurate parameter estimation.

2.3 Local Features

The token-level features used in the system are the same as those used in MSTParser version 0.4.2[3]. The features include lexical forms and (coarse and fine) POS tags of parent tokens, child tokens, their surrounding tokens, and tokens between the child and the parent. The direction and the distance from a parent to its child, and the FEATS fields of the parent and the child which are split into elements and then combined are also included. Features that appeared less than 5 times in training data are ignored.

2.4 Global Features

Global features can capture any information in dependency trees, and the following nine types of global features are used (In the following, *parent node* means a head token, and *child node* means a dependent token):

Child Unigram+Parent+Grandparent This feature template is a 4-tuple consisting of (1) a child node, (2) its parent node, (3) the direction from the parent node to the child node, and (4) the grandparent node.

Each node in the feature template is expanded to its lexical form and coarse POS tag in order to obtain actual features. Features that appeared in four or less sentences are ignored. The same procedure is applied to the following other features.

Child Bigram+Parent This feature template is a 4-tuple consisting of (1) a child node, (2) its parent node, (3) the direction from the parent node to the child node, and (4) the nearest outer sibling node (the nearest sibling node which exists on the opposite side of the parent node) of the child node. This feature template is almost the same as the one used by McDonald and Pereira (2006).

Child Bigram+Parent+Grandparent This feature template is a 5-tuple. The first four elements (1)–(4) are the same as the *Child Bigram+Parent* feature template, and the additional element (5) is the grandparent node.

Child Trigram+Parent This feature template is a 5-tuple. The first four elements (1)–(4) are the same as the *Child Bigram+Parent* feature template, and the additional element (5) is the next nearest outer sibling node of the child node.

Parent+All Children This feature template is a tuple with more than one element. The first element is a parent node, and the other elements are all of its child nodes.

Parent+All Children+Grandparent This feature template is a tuple with more than two elements. The elements other than the last one are the same as the *Parent+All Children* feature template, and the last element is the grandparent node.

Child+Ancestor This feature template is a 2-tuple consisting of (1) a child node, and (2) one of its ancestor nodes.

Acyclic This feature type has one of two values, *true* if the dependency tree is acyclic, or *false* otherwise.

Projective This feature type has one of two values, *true* if the dependency tree is projective, or *false* otherwise.

3 Dependency Relation Labeling

3.1 Model

Dependency relation labeling can be handled as a multi-class classification problem, and we use Support Vector Machines (SVMs) which have been successfully applied to many NLP tasks. Solving large-scale multi-class classification problem with SVMs requires substantial computational resources, so we use the revision learning method (Nakagawa et al., 2002). The revision learning method combines a probabilistic model which has smaller computational cost with a binary classifier which has higher generalization capacity. In the method, the latter classifier revises the output of the former model to conduct multi-class classification with higher accuracy and reasonable computational cost. In this study, we use maximum entropy (ME) models as the probabilistic model and SVMs with the second order polynomial kernel as the binary classifier. The dependency label of each node is determined independently of the labeling of other nodes.

3.2 Features

As the features for SVMs to predict the dependency relation label of the i-th token, we use the lexical forms, coarse and fine POS tags, and FEATS fields of the i-th and the h_i-th tokens. We also use lexical forms and POS tags of the tokens surrounding and in between them (i.e. the j-th token where $j \in \{j | \min\{i, h_i\} - 1 \leq j \leq \max\{i, h_i\} + 1\}$), the grandparent ($h_{h_i}$-th) token, the sibling tokens of i (the j'-th token where $j' \in \{j' | h_{j'} = h_i, j' \neq i\}$),

[3] `http://sourceforge.net/projects/mstparser`

	Arabic	Basque	Catalan	Chinese	Czech	English	Greek	Hungarian	Italian	Turkish	*Average*
LAS	75.08	72.56	87.90	83.84	80.19	88.41	76.31	76.74	83.61	78.22	80.29
UAS	86.09	81.04	92.86	88.88	86.28	90.13	84.08	82.49	87.91	85.77	86.55

Table 1: Results of Multilingual Dependency Parsing

Algorithm	*Features*	Arabic	Basque	Catalan	Chinese	Czech	English	Greek	Hungarian	Italian	Turkish
Eisner	local	85.15	80.20	91.75	86.75	84.19	88.65	83.31	80.27	86.72	84.82
(proj.)	+global	**86.09**	81.00	**92.86**	**88.88**	85.99	**90.13**	**84.08**	81.55	87.91	84.82
CLE	local	84.80	80.39	91.23	86.71	84.21	88.07	83.03	81.15	86.85	85.35
(non-proj.)	+global	85.83	**81.04**	92.64	88.84	**86.28**	90.05	83.87	**82.49**	**87.97**	**85.77**

Table 2: Unlabeled Attachment Scores in Different Settings (underlined values indicate submitted results, and bold values indicate the highest scores)

and the child tokens of i (the j''-th token where $j'' \in \{j''|h_{j''} = i\}$)[4]. As the features for ME models, a subset of them is used since ME models are used just for reducing the search space, and do not need so many features.

4 Results and Analysis

In order to tune the system, we split each training data set into two parts, and used the first half for training and the remaining half for testing in development. The CLE algorithm was used for Basque, Czech, Hungarian and Turkish, and the Eisner algorithm was used for the others. We used lemmas for Catalan, Czech, Greek and Italian, and word forms for all others. The values of the parameters to be fixed were chosen as $R = 500$, $S = 200$, $\sigma = 0.25$, and $\sigma' = 0.25$. With these parameter settings, training took 247 hours, and testing took 343 minutes on an Opteron 250 processor.

Table 1 shows the evaluation results on the test sets. Accuracy was measured with the labeled attachment score (LAS) and the unlabeled attachment score (UAS). Among the participating systems in the shared task, we obtained the second best average accuracy in the labeled attachment score, and the best average accuracy in the unlabeled attachment score. Compared with other systems, the gap between our labeled and unlabeled scores is relatively big. In this study, labeling of dependency relations was performed in a separate post-processing step, and each label was predicted independently. The labeled scores may be improved if the parsing process and the labeling process are performed at the same time, and dependencies among labels are taken into account.

We conducted experiments with different settings. Table 2 shows the results measured with the unlabeled attachment score. In the table, **Eisner** and **CLE** indicate that the Eisner algorithm and the CLE algorithm are used in decoding, and **local** and **+global** indicate that local features alone, and local and global features together are used. The CLE algorithm performed better than the Eisner algorithm for Basque, Czech, Hungarian, Italian and Turkish. All of these data sets except Italian contain relatively a large number of non-projective sentences (the percentage of sentences with at least one non-projective relation in the training data is over 20% (Nivre et al., 2007)), though the Greek data set, on which the Eisner algorithm performed better, also contains many non-projective sentences (20.3%).

By using the global features, the accuracy was improved in all the cases except for Turkish with the Eisner algorithm (Table 2). The increase was rather large in Chinese and Czech. When the global features were used in these languages, the dependency accuracy for tokens whose heads had conjunctions as parts-of-speech was notably improved; from 80.5% to 86.0% in Chinese (Eisner), and from 73.2% to 77.6% in Czech (CLE). We investigated the trained global models, and found that *Parent+All Children* features, whose parents were conjunctions and whose children had compatible classes, had large positive weights, and those whose children had incompatible classes had large negative weights. A feature with a larger weight is generally more influential. Riedel and Clarke (2006) suggested to use linguistic constraints such as "arguments of a coordination must have compatible word classes," and such constraint seemed to be represented by the features in our models.

5 Conclusion

In this study, we applied a dependency parser using global features to multilingual dependency parsing. Evaluation results showed that the use of global features was effective to obtain higher accuracy in multilingual dependency parsing. Improving dependency relation labeling is left for future work.

[4] Although polynomial kernels of SVMs can implicitly handle combined features, some of combined features were also included explicitly because using unnecessarily high order polynomial kernels decreases performance.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque Dependency Treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. 2003. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50:5–43.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

M. Collins and T. Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25–69.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proc. of COLING '96*, pages 340–345.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended Constituent-to-Dependency Conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A Multilingual and Multilevel Annotated Corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proc. of EACL 2006*, pages 81–88.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online Large-Margin Training of Dependency Parsers. In *Proc. of ACL 2005*, pages 91–98.

R. McDonald, F. Pereira, K. Ribarow, and J. Hajic. 2005b. Non-projective dependency parsing using Spanning Tree Algorithms. In *Proc. of HLT/EMNLP 2005*, pages 523–530.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

T. Nakagawa, T. Kudo, and Y. Matsumoto. 2002. Revision Learning and its Application to Part-of-speech Tagging. In *Proc. of ACL 2002*, pages 497–504.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish Treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and Practical Issues in the Construction of a Greek Dependency Treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

S. Riedel and J. Clarke. 2006. Incremental Integer Linear Programming for Non-projective Dependency Parsing. In *Proc. of EMNLP 2006*, pages 129–137.

R. Rosenfeld, S. F. Chen, and X. Zhu. 2001. Whole-Sentence Exponential Language Models: A Vehicle For Linguistic-Statistical Integration. *Computers Speech and Language*, 15(1):55–73.

Experiments with a Higher-Order Projective Dependency Parser

Xavier Carreras
Massachusetts Institute of Technology (MIT)
Computer Science and Artificial Intelligence Laboratory (CSAIL)
32 Vassar St., Cambridge, MA 02139
carreras@csail.mit.edu

Abstract

We present experiments with a dependency parsing model defined on rich factors. Our model represents dependency trees with factors that include three types of relations between the tokens of a dependency and their children. We extend the projective parsing algorithm of Eisner (1996) for our case, and train models using the averaged perceptron. Our experiments show that considering higher-order information yields significant improvements in parsing accuracy, but comes at a high cost in terms of both time and memory consumption. In the multi-lingual exercise of the CoNLL-2007 shared task (Nivre et al., 2007), our system obtains the best accuracy for English, and the second best accuracies for Basque and Czech.

1 Introduction

Structured prediction problems usually involve models that work with factored representations of structures. The information included in the factors determines the type of features that the model can exploit. However, richer representations translate into higher complexity of the inference algorithms associated with the model.

In dependency parsing, the basic *first-order* model is defined by a decomposition of a tree into head-modifier dependencies. Previous work extended this basic model to include *second-order* relations—i.e. dependencies that are adjacent to the main dependency of the factor. Specifically, these approaches considered sibling relations of the modifier token (Eisner, 1996; McDonald and Pereira, 2006). In this paper we extend the parsing model with other types of second-order relations. In particular, we incorporate relations between the head and modifier tokens and the children of the modifier.

One paradigmatic case where the relations we consider are relevant is PP-attachment. For example, in “They sold 1,210 cars in the U.S.”, the ambiguity problem is to determine whether the preposition “in” (which governs “the U.S.”) is modifying “sold” or “cars”, the former being correct in this case. It is generally accepted that to solve the attachment decision it is necessary to look at the head noun within the prepositional phrase (i.e., “U.S.” in the example), which has a grand-parental relation with the two candidate tokens that the phrase may attach—see e.g. (Ratnaparkhi et al., 1994). Other ambiguities in language may also require consideration of grand-parental relations in the dependency structure.

We present experiments with higher-order models trained with averaged perceptron. The second-order relations that we incorporate in the model yield significant improvements in accuracy. However, the inference algorithms for our factorization are very expensive in terms of time and memory consumption, and become impractical when dealing with many labels or long sentences.

2 Higher-Order Projective Models

A dependency parser receives a sentence $\mathbf{x}$ of n tokens, and outputs a labeled dependency tree y. In the tree, a labeled dependency is a triple $\langle h, m, l \rangle$, where $h \in [0 \ldots n]$ is the index of the head token,

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 957–961,
Prague, June 2007.

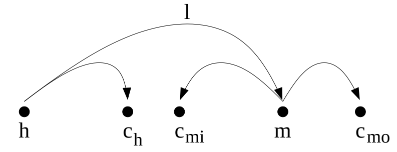


Figure 1: A factor in the higher-order parsing model.

$m \in [1 \dots n]$ is the index of the modifier token, and $l \in [1 \dots L]$ is the label of the dependency. The value $h = 0$ is used for dependencies where the head is a special *root-symbol* of the sentence. We denote by $\mathcal{T}(\mathbf{x})$ the set of all possible dependency structures for a sentence $\mathbf{x}$. In this paper, we restrict to projective dependency trees. The dependency tree computed by the parser for a given sentence is:

$$y^*(\mathbf{x}) = \arg\max_{y \in \mathcal{T}(\mathbf{x})} \sum_{f \in y} \mathrm{score}(\mathbf{w}, \mathbf{x}, f)$$

The parsing model represents a structure y as a set of factors, $f \in y$, and scores each factor using parameters $\mathbf{w}$. In a first-order model a factor corresponds to a single labeled dependency, i.e. $f = \langle h, m, l \rangle$. The features of the model are defined through a feature function $\phi_1(\mathbf{x}, h, m)$ which maps a sentence together with an unlabeled dependency to a feature vector in $\mathbb{R}^{d_1}$. The parameters of the model are a collection of vectors $\mathbf{w}_1^l \in \mathbb{R}^{d_1}$, one for each possible label. The first-order model scores a factor as $\mathrm{score}_1(\mathbf{w}, \mathbf{x}, \langle h, m, l \rangle) = \phi_1(\mathbf{x}, h, m) \cdot \mathbf{w}_1^l$.

The higher-order model defined in this paper decomposes a dependency structure into factors that include children of the head and the modifier. In particular, a factor in our model is represented by the signature $f = \langle h, m, l, c_h, c_{mi}, c_{mo} \rangle$ where, as in the first-order model, h, m and l are respectively the head, modifier and label of the main dependency of the factor; c_h is the child of h in $[h \dots m]$ that is closest to m; c_{mi} is child of m inside $[h \dots m]$ that is furthest from m; c_{mo} is the child of m outside $[h \dots m]$ that is furthest from m. Figure 1 depicts a factor of the higher-order model, and Table 1 lists the factors of an example sentence. Note that a factor involves a main labeled dependency and three adjacent unlabeled dependencies that attach to children of h and m. Special values are used when either of these children are null.

	m	h	c_h	c_{mi}	c_{mo}
They	1	2	-	-	-
sold	2	0	-	1	5
1,200	3	4	-	-	-
cars	4	2	-	3	-
in	5	2	4	-	7
the	6	7	-	-	-
U.S.	7	5	-	6	-

Table 1: Higher-order factors for an example sentence. For simplicity, labels of the factors have been omitted. A first-order model considers only $\langle h, m \rangle$. The second-order model of McDonald and Pereira (2006) considers $\langle h, m, c_h \rangle$. For the PP-attachment decision (factor in row 5), the higher-order model allows us to define features that relate the verb (“sold”) with the content word of the prepositional phrase (“U.S.”).

The higher-order model defines additional second-order features through a function $\phi_2(\mathbf{x}, h, m, c)$ which maps a head, a modifier and a child in a feature vector in $\mathbb{R}^{d_2}$. The parameters of the model are a collection of four vectors for each dependency label: $\mathbf{w}_1^l \in \mathbb{R}^{d_1}$ as in the first-order model; and $\mathbf{w}_{\mathrm{h}}^l, \mathbf{w}_{\mathrm{mi}}^l$ and $\mathbf{w}_{\mathrm{mo}}^l$, all three in $\mathbb{R}^{d_2}$ and each associated to one of the adjacent dependencies in the factor. The score of a factor is:

$$\begin{aligned} &\mathrm{score}_2(\mathbf{w}, \mathbf{x}, \langle h, m, l, c_h, c_{mi}, c_{mo} \rangle) = \\ &\qquad \phi_1(\mathbf{x}, h, m) \cdot \mathbf{w}_1^l + \phi_2(\mathbf{x}, h, m, c_h) \cdot \mathbf{w}_{\mathrm{h}}^l + \\ &\quad \phi_2(\mathbf{x}, h, m, c_{mi}) \cdot \mathbf{w}_{\mathrm{mi}}^l + \phi_2(\mathbf{x}, h, m, c_{mo}) \cdot \mathbf{w}_{\mathrm{mo}}^l \end{aligned}$$

Note that the model uses a common feature function for second-order relations, but features could be defined specifically for each type of relation. Note also that while the higher-order factors include four dependencies, our modelling choice only exploits relations between the main dependency and secondary dependencies. Considering relations between secondary dependencies would greatly increase the cost of the associated algorithms.

2.1 Parsing Algorithm

In this section we sketch an extension of the projective dynamic programming algorithm of Eisner (1996; 2000) for the higher-order model defined above. The time complexity of the algorithm is $O(n^4 L)$, and the memory requirements are $O(n^2 L + n^3)$. As in the Eisner approach, our algorithm visits sentence spans in a bottom up fashion, and constructs a chart with two types of dynamic programming structures, namely open and closed structures—see Figure 2 for a diagram. The dynamic programming structures are:

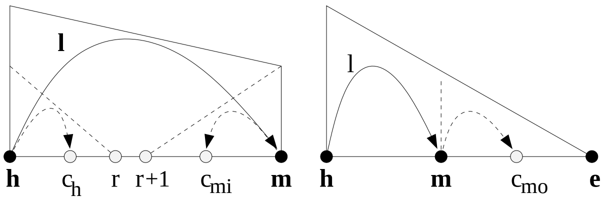


Figure 2: Dynamic programming structures used in the parsing algorithm. The variables in boldface constitute the index of the chart entry for a structure; the other variables constitute the back-pointer stored in the chart entry. Left: an open structure for the chart entry $[h, m, l]_O$; the algorithm looks for the r, c_h and c_{mi} that yield the optimal score for this structure. Right: a closed structure for the chart entry $[h, e, m]_C$; the algorithm looks for the l and c_{mo} that yield the optimal score.

- Open structures: For each span from s to e and each label l, the algorithm maintains a chart entry $[s, e, l]_O$ associated to the dependency $\langle s, e, l \rangle$. For each entry, the algorithm looks for the optimal splitting point r, sibling c_h and grand-child c_{mi} using parameters $\mathbf{w}^l_1$, $\mathbf{w}^l_{\mathrm{h}}$ and $\mathbf{w}^l_{\mathrm{mi}}$. This can be done in $O(n^2)$ because our features do not consider interactions between c_h and c_{mi}. Similar entries $[e, s, l]_O$ are maintained for dependencies headed at e.

- Closed structures: For each span from s to e and each token $m \in [s \dots e]$, the algorithm maintains an entry $[s, e, m]_C$ associated to a partial dependency tree rooted at s in which m is the last modifier of s. The algorithm chooses the optimal dependency label l and grand-child c_{mo} in $O(nL)$, using parameters $\mathbf{w}^l_{\mathrm{mo}}$. Similar entries $[e, s, m]_C$ are maintained for dependencies headed at e.

We implemented two variants of the algorithm. The first forces the root token to participate in exactly one dependency. The second allows many dependencies involving the root token. For the single-root case, it is necessary to treat the root token differently than other tokens. In the experiments, we used the single-root variant if sentences in the training set satisfy this property. Otherwise we used the multi-root variant.

2.2 Features

The first-order features $\phi_1(\mathbf{x}, h, m)$ are the exact same implementation as in previous CoNLL system (Carreras et al., 2006). In turn, those features were inspired by successful previous work in first-order dependency parsing (McDonald et al., 2005). The most basic feature patterns consider the surface form, part-of-speech, lemma and other morpho-syntactic attributes of the head or the modifier of a dependency. The representation also considers complex features that exploit a variety of conjunctions of the forms and part-of-speech tags of the following items: the head and modifier; the head, modifier, and any token in between them; the head, modifier, and the two tokens following or preceding them.

As for the second-order features, we again base our features with those of McDonald and Pereira (2006), who reported successful experiments with second-order models. We add some patterns to their features. Let dir be "right" if $h < m$, and "left" otherwise; let $\mathrm{form}(x_i)$ and $\mathrm{cpos}(x_i)$ return the surface form and coarse part-of-speech of token x_i, respectively. The definition of $\phi_2(\mathbf{x}, h, m, c)$ is:

- $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_m) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{cpos}(x_m) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{form}(x_h) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{form}(x_m) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{cpos}(x_h) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{cpos}(x_m) \cdot \mathrm{form}(x_c)$
- $dir \cdot \mathrm{form}(x_h) \cdot \mathrm{cpos}(x_c)$
- $dir \cdot \mathrm{form}(x_m) \cdot \mathrm{cpos}(x_c)$

3 Experiments and Results

We report experiments with higher-order models for the ten languages in the multilingual track of the CoNLL-2007 shared task (Nivre et al., 2007).[1]

In all experiments, we trained our models using the averaged perceptron (Freund and Schapire, 1999), following the extension of Collins (2002) for structured prediction problems. To train models, we used "projectivized" versions of the training dependency trees.[2]

[1] We are grateful to the providers of the treebanks that constituted the data for the shared task (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003).

[2] We obtained projective trees for training sentences by running the projective parser with an oracle model (that assigns a score of +1 to correct dependencies and -1 otherwise).

	Catalan	Czech	English
First-Order, no averaging	82.07	68.98	83.75
First-Order	86.15	75.96	87.54
Higher-Order, c_h	87.50	77.15	88.70
Higher-Order, c_h c_{mo}	87.68	77.62	89.28
Higher-Order, c_h c_{mi} c_{mo}	88.04	78.09	89.59

Table 2: Labeled attachment scores on validation data (∼10,000 tokens per language), for different models that exploit increasing orders of factorizations.

3.1 Impact of Higher-Order Factorization

Our first set of experiments looks at the performance of different factorizations. We selected three languages with a large number of training sentences, namely Catalan, Czech and English. To evaluate models, we held out the training sentences that cover the first 10,000 tokens; the rest was used for training.

We compared four models at increasing orders of factorizations. The first is a first-order model. The second model is similar to that of McDonald and Pereira (2006): a factor consists of a main labeled dependency and the head child closest to the modifier (c_h). The third model incorporates the modifier child outside the main dependency in the factorization (c_{mo}). Finally, the last model incorporates the modifier child inside the dependency span (c_{mi}), thus corresponding to the complete higher-order model presented in the previous section.

Table 2 shows the accuracies of the models on validation data. Each model was trained for up to 10 epochs, and evaluated at the end of each epoch; we report the best accuracy of these evaluations. Clearly, the accuracy increases as the factors include richer information in terms of second-order relations. The richest model obtains the best accuracy in the three languages, being much better than that of the first-order model. The table also reports the accuracy of an unaveraged first-order model, illustrating the benefits of parameter averaging.

3.2 Results on the Multilingual Track

We trained a higher-order model for each language, using the averaged perceptron. In the experiments presented above we observed that the algorithm does not over-fit, and that after two or three training epochs only small variations in accuracy occur. Based on this fact, we designed a criterion to train models: we ran the training algorithm for up to three days of computation, or a maximum of 15 epochs. For Basque, Chinese and Turkish we could complete the 15 epochs. For Arabic and Catalan, we could only complete 2 epochs. Table 3 reports the performance of the higher-order projective models on the ten languages of the multilingual track.

	training		test	
	sent./min.	mem.	UAS	LAS
Arabic	1.21	1.8GB	81.48	70.20
Basque	33.15	1.2GB	81.08	75.73
Catalan	5.50	1.7GB	92.46	87.60
Chinese	1461.66	60MB	86.20	80.86
Czech	18.19	1.8GB	85.16	78.60
English	15.57	1.0GB	90.63	89.61
Greek	8.10	250MB	81.37	73.56
Hungarian	5.65	1.6GB	79.92	75.42
Italian	12.44	900MB	87.19	83.46
Turkish	116.55	600MB	82.41	75.85
Average	-	-	84.79	79.09

Table 3: Performance of the higher-order projective models on the multilingual track of the CoNLL-2007 task. The first two columns report the speed (in sentences per minute) and memory requirements of the training algorithm—these evaluations were made on the first 1,000 training sentences with a Dual-Core AMD Opteron$^{\mathrm{TM}}$ Processor 256 at 1.8GHz with 4GB of memory. The last two columns report unlabelled (UAS) and labelled (LAS) attachment scores on test data.

4 Conclusion

We have presented dependency parsing models that exploit higher-order factorizations of trees. Such factorizations allow the definition of second-order features associated with sibling and grand-parental relations. For some languages, our models obtain state-of-the-art results.

One drawback of our approach is that the inference algorithms for higher-order models are very expensive. For languages with many dependency labels or long sentences, training and parsing becomes impractical for current machines. Thus, a promising line of research is the investigation of methods to efficiently incorporate higher-order relations in discriminative parsing.

Acknowledgments

I am grateful to Terry Koo, Amir Globerson and Michael Collins for their helpful comments relating this work, and to the anonymous reviewers for their suggestions. A significant part of the system and the code was based on my previous system in the CoNLL-X task, developed with Mihai Surdeanu and Lluís Màrquez at the UPC. The author was supported by the Catalan Ministry of Innovation, Universities and Enterprise.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

X. Carreras, M. Surdeanu, and L. Màrquez. 2006. Projective dependency parsing with perceptron. In *Proc. CoNLL-X*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP-2002*.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H. C. Bunt and A. Nijholt, editors, *New Developments in Natural Language Parsing*, pages 29–62. Kluwer Academic Publishers.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL-2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. ACL*.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of EMNLP-CoNLL*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

A. Ratnaparkhi, J. Reinar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proc. of the ARPA Workshop on Human Language Technology*.

Log-linear Models of Non-projective Trees, k-best MST Parsing and Tree-ranking

Keith Hall[1] and **Jiří Havelka**[2] and **David A. Smith**[1]

[1]Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD USA
keith_hall@jhu.edu
dasmith@cs.jhu.edu

[2]Institute of Formal and Applied Linguistics
Charles University
Prague, Czech Republic
havelka@ufal.mff.cuni.cz

Abstract

We present our system used in the CoNLL 2007 shared task on multilingual parsing. The system is composed of three components: a k-best maximum spanning tree (MST) parser, a tree labeler, and a reranker that orders the k-best labeled trees. We present two techniques for training the MST parser: tree-normalized and graph-normalized conditional training. The tree-based reranking model allows us to explicitly model *global* syntactic phenomena. We describe the reranker features which include non-projective edge attributes. We provide an analysis of the errors made by our system and suggest changes to the models and features that might rectify the current system.

1 Introduction

Reranking the output of a k-best parser has been shown to improve upon the best results of a state-of-the-art constituency parser (Charniak and Johnson, 2005). This is primarily due to the ability to incorporate complex structural features that cannot be modeled under a CFG. Recent work shows that k-best maximum spanning tree (MST) parsing and reranking is also viable (Hall, 2007). In the current work, we explore the k-best MST parsing paradigm along with a tree-based reranker. A system using the parsing techniques presented in this paper was entered in the CoNLL 2007 shared task competition (Nivre et al., 2007). This task evaluated parsing performance on 10 languages: Arabic, Basque, Catalan, Chinese, Czech, English, Greek, Hungarian, Italian, and Turkish using data originating from a wide variety of dependency treebanks, and transformations of constituency-based treebanks (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003).

We show that oracle parse accuracy[1] of the output of our k-best parser is generally higher than the best reported results. We also present the results of a reranker based on a rich set of structural features, including features explicitly targeted at modeling non-projective configurations. Labeling of the dependency edges is accomplished by an edge labeler based on the same feature set as used in training the k-best MST parser.

2 Parser Description

Our parser is composed of three components: a k-best MST parser, a tree-labeler, and a tree-reranker. Log-linear models are used for each of the components independently. In this section we give an overview of the models, the training techniques, and the decoders.

2.1 MST Parsing, Reranking, and Labeling

The connection between the maximum spanning tree problem and dependency parsing stems from the observation that a dependency parse is simply an oriented spanning tree on the graph of all possible

[1] The oracle accuracy for a set of hypotheses is the maximal accuracy for any of the hypotheses.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 962–966,
Prague, June 2007.

dependency links (the fully connected dependency graph). Unfortunately, by mapping the problem to a graph, we assume that the scores associated with edges are independent, and thus, are limited to *edge-factored* models.

Edge-factored models are severely limited in their capacity to predict structure. In fact, they can only directly model parent-child links. In order to alleviate this, we use a k-best MST parser to generate a set of candidate hypotheses. Then, we rerank these trees using a model based on rich structural features that model features such as valency, subcategorization, ancestry relationships, and sibling interactions, as well as features capturing the global structure of dependency trees, aimed primarily at modeling language specific non-projective configurations.

We assign dependency labels to entire trees, rather than predicting the labels during tree construction. Given that we have a reranking process, we can label the k-best tree hypotheses output from our MST parser, and rerank the labeled trees. We have explored both labeled and unlabeled reranking. In the latter case, we simply label the maximal unlabeled tree.

2.1.1 MST Training

McDonald et al. (2005) present a technique for training discriminative models for dependency parsing. The edge-factored models we use for MST parsing are closely related to those described in the previous work, but allow for the efficient computation of normalization factors which are required for first and second-order (gradient-based) training techniques.

We consider two estimation procedures for *parent-prediction* models. A parent-prediction model assigns a conditional score $s(g|d)$ for every parent-child pair (we denote the parent/governor g, and the child/dependent d), where $s(g|d) = s(g,d)/\sum_{g'} s(g',d)$. In our work, we compute probabilities $p(g|d)$ based on conditional log-linear models. This is an approximation to a generative model that predicts each node once (i.e., $\prod_d p(d|g)$).

In the graph-normalized model, we assume that the conditional distributions are independent of one another. In particular, we find the model parameters that maximize the likelihood of $p(g^*|d)$, where g^* is the correct parent in the training data. We perform the optimization over the entire training set, tying the feature parameters. In particular, we perform maximum entropy (MaxEnt) estimation over the conditional distribution using second-order gradient descent optimization techniques.[2] An advantage of the parent-prediction model is that we can frame the estimation problem as that of minimum-error training with a zero-one loss term:

$$p(e,g|d) = \frac{\exp(\sum_i \lambda_i f_i(e,g,d))}{Z_d} \qquad (1)$$

where $e \in \{0,1\}$ is the error term (e is 1 for the correct parent and 0 for all other nodes) and $Z_d = \sum_j \exp(\sum_i \lambda_i f_i(e_j, g_j, d))$ is the normalization constant for node d. Note that the normalization factor considers all graphs with in-degree zero for the root node and in-degree one for other nodes.

At parsing time, of course, our parent predictions are constrained to produce a (non-projective) tree structure. We can sum over all non-projective spanning trees by taking the determinant of the Kirchhoff matrix of the graph defined above, minus the row and column corresponding to the root node (Smith and Smith, 2007). Training graph-normalized and tree-normalized models under identical conditions, we find tree normalization wins by 0.5% to 1% absolute dependency accuracy. Although tree normalization also shows a (smaller) advantage in k-best oracle accuracy, we do not believe it would have a large effect on our reranking results.

2.1.2 Reranker Training

The reranker is based on a conditional log-linear model subject to the MaxEnt constraints using the same second-order optimization procedures as the graph-normalized MST models. The primary difference here is that there is no single correct tree in the set of k candidate parse trees. Instead, we have k trees that are generated by our k-best parser, each with a score assigned by the parser. If we are performing labeled reranking, we label each of these hypotheses with l possible labelings, each with a score assigned by the labeler.

As with the parent-prediction, graph-normalized model, we perform minimum-error training. The

[2] For the graph-normalized models, we use L-BFGS optimization provided through the TAO/PETSC optimization library (Benson et al., 2005; Balay et al., 2004).

optimization is achieved by assuming the oracle-best parse(s) are correct and the remaining hypotheses are incorrect. Furthermore, the feature values are scaled according to the relative difference between the oracle-best score and the score assigned to the non-oracle-best hypothesis.

Note that any reranker could be used in place of our current model. We have chosen to keep the reranker model closely related to the MST parsing model so that we can share feature representations and training procedures.

2.1.3 Labeler Training

We used the same edge features to train a separate log-linear labeling model. Each edge feature was conjoined with a potential label, and we then maximized the likelihood of the labeling in the training data. Since this model is also edge-factored, we can store the labeler scores for each of the n^2 potential edges in the dependency tree. In the submitted system, we simply extracted the Viterbi predictions of the labeler for the unlabeled trees selected by the reranker. We also (see below) ran experiments where each entry in the k-best lists input as training data to the reranker was augmented by its l-best labelings. We hoped thereby to inject more diversity into the resulting structures.

2.1.4 Model Features

Our MST models are based on the features described in (Hall, 2007); specifically, we use features based on a dependency nodes' form, lemma, coarse and fine part-of-speech tag, and morphological-string attributes. Additionally, we use surface-string distance between the parent and child, buckets of features indicating if a particular form/lemma/tag occurred between or next to the parent and child, and a branching feature indicating whether the child is to the left or right of the parent. Composite features, combining the above features are also included (e.g., a single feature combining branching, parent & child form, parent & child tag).

The tree-based reranker includes the features described in (Hall, 2007) as well as features based on non-projective edge attributes explored in (Havelka, 2007a; Havelka, 2007b). One set of features models relationships of nodes with their siblings, including valency and subcategorization. A second set of features models global tree structure and includes features based on a node's ancestors and the depth and size of its subtree. A third set of features models the interaction of word order and tree structure as manifested on individual edges, i.e., the features model language specific projective and non-projective configurations. They include edge-based features corresponding to the global constraints of projectivity, planarity and well-nestedness, and for non-projective edges, they furthermore include level type, level signature and ancestor-in-gap features. All features allow for an arbitrary degree of lexicalization; in the reported results, the first two sets of features use coarse and fine part-of-speech lexicalizations, while the features in the third set are used in their unlexicalized form due to time limitations.

3 Results and Analysis

Hall (2007) shows that the oracle parsing accuracy of a k-best edge-factored MST parser is considerably higher than the one-best score of the same parser, even when k is small. We have verified that this is true for the CoNLL shared-task data by evaluating the oracle rates on a randomly sampled development set for each language.

In order to select optimal model parameters for the MST parser, the labeler, and reranker, we sampled approximately 200 sentences from each training set to use as a development test set. Training the reranker requires a jackknife n-fold training procedure where $n-1$ partitions are used to train a model that parses the remaining partition. This is done n times to generate k-best parses for the entire training set without using models trained on the data they are run on.

For lack of space, we report only results on the CoNLL evaluation data set here, but note that the trends observed on the evaluation data are identical to those observed on our development sets.

In Table 1 we present results for labeled (and unlabeled) dependency accuracy on the CoNLL 2007 evaluation data set. We report the oracle accuracy for different sized k-best hypothesis sets. The columns are labeled by the number of trees output from the MST parser, k;[3] and by the number of al-

[3] All results are reported for the graph-normalized training technique.

Language	**Oracle Accuracy**				**New**	CoNLL07	CoNLL07
	$k = 1, l = 1$	$k = 10, l = 5$	$k = 50, l = 1$	$k = 50, l = 2$	Reranked	Reported	Best
Arabic	(83.10)	(85.56)	(86.96)		(83.67)	73.40 (83.45)	76.52 (86.09)
Basque	67.92 (76.88)	76.25 (82.19)	69.93 (84.99)	76.81	(77.76)	69.80 (78.52)	76.92 (82.80)
Catalan	82.28 (87.82)	85.11 (90.87)	86.82 (92.68)	86.82	(89.43)	82.38 (87.80)	88.70 (93.40)
Chinese	73.86 (85.58)	91.32 (93.39)	82.39 (95.80)	92.21	(87.87)	82.77 (87.91)	84.69 (88.94)
Czech	74.05 (80.21)	78.58 (85.08)	80.97 (87.60)	80.97	(82.20)	72.27 (78.47)	80.19 (86.28)
English	82.21 (83.63)	85.95 (87.59)	87.99 (89.75)	87.99	(85.31)	81.93 (83.21)	89.61 (90.63)
Greek	72.21 (81.16)	78.58 (84.89)	74.13 (86.95)	79.48	(81.81)	74.21 (82.04)	76.31 (84.08)
Hungarian	71.68 (78.57)	79.70 (83.03)	74.32 (85.12)	80.75	(80.05)	74.20 (79.34)	80.27 (83.55)
Italian	77.92 (83.16)	85.05 (87.54)	80.30 (89.66)	86.42	(84.71)	80.69 (84.81)	84.40 (87.91)
Turkish	75.34 (83.63)	83.96 (89.65)	77.78 (92.40)	84.98	(84.13)	77.42 (85.18)	79.81 (86.22)

Table 1: Labeled (unlabeled) attachment accuracy for k-best MST oracle results and reranked data on the evaluation set. The 1-best results ($k = 1, l = 1$) represent the performance of the MST parser without reranking. The *New Reranked* field shows recent unlabeled reranking results of 50-best trees using a modified feature set. For arabic, we only report unlabeled accuracy for different k and l.

ternative labelings for each tree, l. When $k = 1$, the score is the best achievable by the edge-factored MST parser using our models. As k increases, the oracle parsing accuracy increases. The most extreme difference between the one-best accuracy and the 50-best oracle accuracy can be seen for Turkish where there is a difference of 9.64 points of accuracy (8.77 for the unlabeled trees). This means that the reranker need only select the correct tree from a set of 50 to increase the score by 9.64%. As our reranking results show, this is not as simple as it may appear.

We report the results for our CoNLL submission as well as recent results based on alternative parameters optimization on the development set. We report the latest results only for unlabeled accuracy of reranking 50-best MST output.

4 Conclusion

Our submission to the CoNLL 2007 shared task on multilingual parsing supports the hypothesis that edge-factored MST parsing is viable given an effective reranker. The reranker used in our submission was unable to achieve the oracle rates. We believe this is primarily related to a relatively impoverished feature set. Due to time constraints, we have not been able to train lexicalized reranking models. The introduction of lexicalized features in the reranker should influence the selection of better trees, which we know exist in the k-best hypothesis sets.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. 2004. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory.

Steven J. Benson, Lois Curfman McInnes, Jorge Moré, and Jason Sarich. 2005. TAO user manual (revision 1.8). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory. http://www.mcs.anl.gov/tao.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

Keith Hall. 2007. k-best spanning tree parsing. In *(To Appear) Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Jiří Havelka. 2007a. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *(To Appear) Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Jiří Havelka. 2007b. Relationship between non-projective edges, their level types, and well-nestedness. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 61–64.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *(To Appear) Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing*.

Improving Translation Quality by Discarding Most of the Phrasetable

J Howard Johnson and **Joel Martin**
Interactive Information Group
National Research Council Canada
Ottawa, Ontario, Canada
firstname.lastname@nrc.gc.ca

George Foster and **Roland Kuhn**
Interactive Language Technologies Group
National Research Council Canada
Gatineau, Québec, Canada
firstname.lastname@nrc.gc.ca

Abstract

It is possible to reduce the bulk of phrasetables for Statistical Machine Translation using a technique based on the significance testing of phrase pair co-occurrence in the parallel corpus. The savings can be quite substantial (up to 90%) and cause no reduction in BLEU score. In some cases, an improvement in BLEU is obtained at the same time although the effect is less pronounced if state-of-the-art phrasetable smoothing is employed.

1 Introduction

An important part of the process of Statistical Machine Translation (SMT) involves inferring a large table of phrase pairs that are translations of each other from a large corpus of aligned sentences. These phrase pairs together with estimates of conditional probabilities and useful feature weights, called collectively a *phrasetable*, are used to match a source sentence to produce candidate translations. The choice of the best translation is made based on the combination of the probabilities and feature weights, and much discussion has been made of how to make the estimates of probabilites, how to smooth these estimates, and what features are most useful for discriminating among the translations.

However, a cursory glance at phrasetables produced often suggests that many of the translations are wrong or will never be used in any translation. On the other hand, most obvious ways of reducing the bulk usually lead to a reduction in translation quality as measured by BLEU score. This has led to an impression that these pairs must contribute something in the grand scheme of things and, certainly, more data is better than less.

Nonetheless, this bulk comes at a cost. Large tables lead to large data structures that require more resources and more time to process and, more importantly, effort directed in handling large tables could likely be more usefully employed in more features or more sophisticated search.

In this paper, we show that it is possible to prune phrasetables using a straightforward approach based on significance testing, that this approach does not adversely affect the quality of translation as measured by BLEU score, and that savings in terms of number of discarded phrase pairs can be quite substantial. Even more surprising, pruning can actually raise the BLEU score although this phenomenon is less prominent if state of the art smoothing of phrasetable probabilities is employed.

Section 2 reviews the basic ideas of Statistical Machine Translation as well as those of testing significance of associations in two by two contingency tables departing from independence. From this, a filtering algorithm will be described that keeps only phrase pairs that pass a significance test. Section 3 outlines a number of experiments that demonstrate the phenomenon and measure its magnitude. Section 4 presents the results of these experiments. The paper concludes with a summary of what has been learned and a discussion of continuing work that builds on these ideas.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 967–975, Prague, June 2007.

2 Background Theory

2.1 Our Approach to Statistical Machine Translation

We define a phrasetable as a set of source phrases (n-grams) $\tilde{s}$ and their translations (m-grams) $\tilde{t}$, along with associated translation probabilities $p(\tilde{s}|\tilde{t})$ and $p(\tilde{t}|\tilde{s})$. These conditional distributions are derived from the joint frequencies $c(\tilde{s}, \tilde{t})$ of source / target n, m-grams observed in a word-aligned parallel corpus. These joint counts are estimated using the phrase induction algorithm described in (Koehn et al., 2003), with symmetrized word alignments generated using IBM model 2 (Brown et al., 1993). Phrases are limited to 8 tokens in length ($n, m \leq 8$).

Given a source sentence $\mathbf{s}$, our phrase-based SMT system tries to find the target sentence $\hat{\mathbf{t}}$ that is the most likely translation of $\mathbf{s}$. To make search more efficient, we use the Viterbi approximation and seek the most likely combination of $\mathbf{t}$ and its alignment $\mathbf{a}$ with $\mathbf{s}$, rather than just the most likely $\mathbf{t}$:

$$\hat{\mathbf{t}} = \operatorname*{argmax}_{\mathbf{t}} p(\mathbf{t}|\mathbf{s}) \approx \operatorname*{argmax}_{\mathbf{t},\mathbf{a}} p(\mathbf{t}, \mathbf{a}|\mathbf{s}),$$

where $\mathbf{a} = (\tilde{s}_1, \tilde{t}_1, j_1), ..., (\tilde{s}_K, \tilde{t}_K, j_K)$; $\tilde{t}_k$ are target phrases such that $\mathbf{t} = \tilde{t}_1 ... \tilde{t}_K$; $\tilde{s}_k$ are source phrases such that $\mathbf{s} = \tilde{s}_{j_1} ... \tilde{s}_{j_K}$; and $\tilde{s}_k$ is the translation of the kth target phrase $\tilde{t}_k$.

To model $p(\mathbf{t}, \mathbf{a}|\mathbf{s})$, we use a standard loglinear approach:

$$p(\mathbf{t}, \mathbf{a}|\mathbf{s}) \propto \exp \left[\sum_i \lambda_i f_i(\mathbf{s}, \mathbf{t}, \mathbf{a}) \right]$$

where each $f_i(\mathbf{s}, \mathbf{t}, \mathbf{a})$ is a feature function, and weights λ_i are set using Och's algorithm (Och, 2003) to maximize the system's BLEU score (Papineni et al. , 2001) on a development corpus. The features used are: the length of $\mathbf{t}$; a single-parameter distortion penalty on phrase reordering in a, as described in (Koehn et al., 2003); phrase translation model probabilities; and 4-gram language model probabilities $\log p(\mathbf{t})$, using Kneser-Ney smoothing as implemented in the SRILM toolkit (Stolcke, 2002).

Phrase translation model probabilities are features of the form:

$$\log p(\mathbf{s}|\mathbf{t}, \mathbf{a}) \approx \sum_{k=1}^{K} \log p(\tilde{s}_k|\tilde{t}_k)$$

i.e., we assume that the phrases $\tilde{s}_k$ specified by $\mathbf{a}$ are conditionally independent, and depend only on their aligned phrases $\tilde{t}_k$.

The "forward" phrase probabilities $p(\tilde{t}|\tilde{s})$ are not used as features, but only as a filter on the set of possible translations: for each source phrase $\tilde{s}$ that matches some ngram in $\mathbf{s}$, only the 30 top-ranked translations $\tilde{t}$ according to $p(\tilde{t}|\tilde{s})$ are retained. One of the reviewers has pointed out correctly that taking only the top 30 translations will interact with the subject under study; however, this pruning technique has been used as a way of controlling the width of our beam search and rebalancing search parameters would have complicated this study and taken it away from our standard practice.

The phrase translation model probabilities are smoothed according to one of several techniques as described in (Foster et al., 2006) and identified in the discussion below.

2.2 Significance testing using two by two contingency tables

Each phrase pair can be thought of as am n, m-gram $(\tilde{s}, \tilde{t})$ where $\tilde{s}$ is an n-gram from the source side of the corpus and $\tilde{t}$ is an m-gram from the target side of the corpus.

We then define: $C(\tilde{s}, \tilde{t})$ as the number of parallel sentences that contain one or more occurrences of $\tilde{s}$ on the source side and $\tilde{t}$ on the target side; $C(\tilde{s})$ the number of parallel sentences that contain one or more occurrences of $\tilde{s}$ on the source side; and $C(\tilde{t})$ the number of parallel sentences that contain one or more occurrences of $\tilde{t}$ on the target side. Together with N, the number of parallel sentences, we have enough information to draw up a two by two contingency table representing the unconditional relationship between $\tilde{s}$ and $\tilde{t}$. This table is shown in Table 1.

A standard statistical technique used to assess the importance of an association represented by a contingency table involves calculating the probability that the observed table or one that is more extreme could occur by chance assuming a model of independence. This is called a significance test. Introductory statistics texts describe one such test called the Chi-squared test.

There are other tests that more accurately apply to our small tables with only two rows and columns.

Table 1: Two by two contingency table for $\tilde{s}$ and $\tilde{t}$

$C(\tilde{s}, \tilde{t})$	$C(\tilde{s}) - C(\tilde{s}, \tilde{t})$	$C(\tilde{s})$
$C(\tilde{t}) - C(\tilde{s}, \tilde{t})$	$N - C(\tilde{s}) - C(\tilde{t}) + C(\tilde{s}, \tilde{t})$	$N - C(\tilde{s})$
$C(\tilde{t})$	$N - C(\tilde{t})$	N

In particular, Fisher's exact test calculates probability of the observed table using the hypergeometric distibution.

$$p_h(C(\tilde{s}, \tilde{t})) = \frac{\binom{C(\tilde{s})}{C(\tilde{s}, \tilde{t})}\binom{N - C(\tilde{s})}{C(\tilde{t}) - C(\tilde{s}, \tilde{t})}}{\binom{N}{C(\tilde{t})}}$$

The p-value associated with our observed table is then calculated by summing probabilities for tables that have a larger $C(\tilde{s}, \tilde{t})$).

$$\text{p-value}(C(\tilde{s}, \tilde{t})) = \sum_{k=C(\tilde{s}, \tilde{t})}^{\infty} p_h(k)$$

This probability is interpreted as the probability of observing by chance an association that is at least as strong as the given one and hence its significance. Agresti (1996) provides an excellent introduction to this topic and the general ideas of significance testing in contingency tables.

Fisher's exact test of significance is considered a gold standard since it represents the precise probabilities under realistic assumptions. Tests such as the Chi-squared test or the log-likelihood-ratio test (yet another approximate test of significance) depend on asymptotic assumptions that are often not valid for small counts.

Note that the count $C(\tilde{s}, \tilde{t})$ can be larger or smaller than $c(\tilde{s}, \tilde{t})$ discussed above. In most cases, it will be larger, because it counts all co-occurrences of $\tilde{s}$ with $\tilde{t}$ rather than just those that respect the word alignment. It can be smaller though because multiple co-occurrences can occur within a single aligned sentence pair and be counted multiple times in $c(\tilde{s}, \tilde{t})$. On the other hand, $C(\tilde{s}, \tilde{t})$ will not count all of the possible ways that an n, m-gram match can occur within a single sentence pair; it will count the match only once per sentence pair in which it occurs.

Moore (2004) discusses the use of significance testing of word associations using the log-likelihood-ratio test and Fisher's exact test. He shows that Fisher's exact test is often a practical method if a number of techniques are followed:

1. approximating the logarithms of factorials using commonly available numerical approximations to the log gamma function,

2. using a well-known recurrence for the hypergeometic distribution,

3. noting that few terms usually need to be summed, and

4. observing that convergence is usually rapid.

2.3 Significance pruning

The idea behind significance pruning of phrasetables is that not all of the phrase pairs in a phrasetable are equally supported by the data and that many of the weakly supported pairs could be removed because:

1. the chance of them occurring again might be low, and

2. their occurrence in the given corpus may be the result of an artifact (a combination of effects where several estimates artificially compensate for one another). This concept is usually referred to as overfit since the model fits aspects of the training data that do not lead to improved prediction.

Phrase pairs that cannot stand on their own by demonstrating a certain level of significance are suspect and removing them from the phrasetable may be beneficial in terms of reducing the size of data structures. This will be shown to be the case in rather general terms.

Note that this pruning may and quite often will remove all of the candidate translations for a source phrase. This might seem to be a bad idea but it must be remembered that deleting longer phrases will allow combinations of shorter phrases to be used and these might have more and better translations from the corpus. Here is part of the intuition about how phrasetable smoothing may interact with phrasetable pruning: both are discouraging longer but infrequent phrases from the corpus in favour of combinations of more frequent, shorter phrases.

Because the probabilities involved below will be so incredibly tiny, we will work instead with the negative of the natural logs of the probabilities. Thus instead of selecting phrase pairs with a p-value less than $\exp(-20)$, we will select phrase pairs with a negative-log-p-value greater than 20. This has the advantage of working with ordinary-sized numbers and the happy convention that bigger means more pruning.

2.4 $C(\tilde{s}, \tilde{t}) = 1$, 1-1-1 Tables and the α Threshold

An important special case of a table occurs when a phrase pair occurs exactly once in the corpus, and each of the component phrases occurs exactly once in its side of the parallel corpus.

These phrase pairs will be referred to as 1-1-1 phrase pairs and the corresponding tables will be called 1-1-1 contingency tables because $C(\tilde{s}) = 1$, $C(\tilde{t}) = 1$, and $C(\tilde{s}, \tilde{t}) = 1$.

Moore (2004) comments that the p-value for these tables under Fisher's exact test is $1/N$. Since we are using thresholds of the negative logarithm of the p-value, the value $\alpha = log(N)$ is a useful threshold to consider.

In particular, $\alpha + \epsilon$ (where ϵ is an appropriately small positive number) is the smallest threshold that results in none of the 1-1-1 phrase pairs being included. Similarly, $\alpha - \epsilon$ is the largest threshold that results in all of the 1-1-1 phrase pairs being included. Because 1-1-1 phrase pairs can make up a large part of the phrase table, this is important observation for its own sake.

Since the contingency table with $C(\tilde{s}, \tilde{t}) = 1$ having the greatest significance (lowest p-value) is the 1-1-1 table, using the threshold of $\alpha + \epsilon$ can be used to exclude all of the phrase pairs occurring exactly once ($C(\tilde{s}, \tilde{t}) = 1$).

The common strategy of deleting all of the 1-count phrase pairs is very similar in effect to the use of the $\alpha + \epsilon$ threshold.

3 Experiments

3.1 WMT06

The corpora used for most of these experiments are publicly available and have been used for a number of comparative studies (Workshop on Statistical Machine Translation, 2006). Provided as part of the materials for the shared task are parallel corpora for French–English, Spanish–English, and German–English as well as language models for English, French, Spanish, and German. These are all based on the Europarl resources (Europarl, 2003).

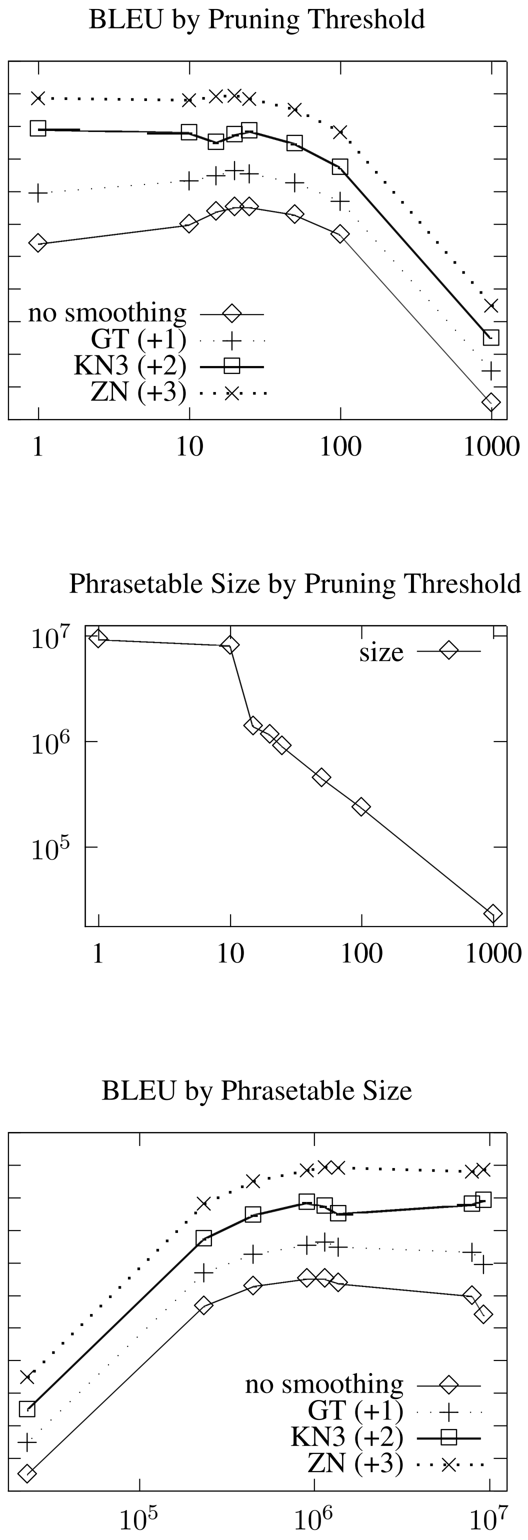
The only change made to these corpora was to convert them to lowercase and to Unicode UTF-8. Phrasetables were produced by symmetrizing IBM2 conditional probabilities as described above.

The phrasetables were then used as a list of n, m-grams for which counts $C(\tilde{s}, \tilde{t})$, $C(\tilde{s})$, and $C(\tilde{t})$ were obtained. Negative-log-p-values under Fisher's exact test were computed for each of the phrase pairs in the phrasetable and the entry was censored if the negative-log-p-value for the test was below the pruning threshold. The entries that are kept are ones that are highly significant.

A number of combinations involving many different pruning thresholds were considered: no pruning, 10, $\alpha - \epsilon$, $\alpha + \epsilon$, 15, 20, 25, 50, 100, and 1000. In addition, a number of different phrasetable smoothing algorithms were used: no smoothing, Good-Turing smoothing, Kneser-Ney 3 parameter smoothing and the loglinear mixture involving two features called Zens-Ney (Foster et al., 2006).

3.2 Chinese

To test the effects of significance pruning on larger corpora, a series of experiments was run on a much larger corpus based on that distributed for MT06 Chinese–English (NIST MT, 2006). Since the objective was to assess how the method scaled we used our preferred phrasetable smoothing technique of


BLEU by Pruning Threshold

Phrasetable Size by Pruning Threshold

BLEU by Phrasetable Size

Figure 1: WMT06: Results for French ⟶ English. [to separate the curves, graphs for smoothed methods are shifted by +1, +2, or +3 BLEU points]

Table 2: Corpus Sizes and α Values

	number of parallel sentences	α
WMT06: fr ⟷ en	688,031	13.4415892
WMT06: es ⟷ en	730,740	13.501813
WMT06: de ⟷ en	751,088	13.5292781
Chinese–English: best	3,164,228	14.9674197
Chinese–English: UN-v2	4,979,345	15.4208089

Zens-Ney and separated our corpus into two phrasetables, one based on the UN corpus and the other based on the best of the remaining parallel corpora available to us.

Different pruning thresholds were considered: no pruning, 14, 16, 18, 20, and 25. In addition, another more aggressive method of pruning was attempted. Moore points out, correctly, that phrase pairs that occur in only one sentence pair, ($C(\tilde{s}, \tilde{t}) = 1$), are less reliable and might require more special treatment. These are all pruned automatically at thresholds of 16 and above but not at threshold of 14. A special series of runs was done for threshold 14 with all of these singletons removed to see whether at these thresholds it was the significance level or the pruning of phrase pairs with ($C(\tilde{s}, \tilde{t}) = 1$) that was more important. This is identified as $14'$ in the results.

4 Results

The results of the experiments are described in Tables 2 through 6.

Table 2 presents the sizes of the various parallel corpora showing the number of parallel sentences, N, for each of the experiments, together with the α thresholds ($\alpha = log(N)$).

Table 3 shows the sizes of the phrasetables that result from the various pruning thresholds described for the WMT06 data. It is clear that this is extremely aggressive pruning at the given levels.

Table 4 shows the corresponding phrasetable sizes for the large corpus Chinese–English data. The pruning is not as aggressive as for the WMT06 data but still quite sizeable.

Tables 5 and 6 show the main results for the WMT06 and the Chinese–English large corpus experiments. To make these results more graphic, Figure 1 shows the $French \longrightarrow English$ data from the WMT06 results in the form of three graphs. Note

Table 3: WMT06: Distinct phrase pairs by pruning threshold

threshold	fr ⟷ en		es ⟷ en		de ⟷ en	
none	9,314,165	100%	11,591,013	100%	6,954,243	100%
10	7,999,081	85.9%	10,212,019	88.1%	5,849,593	84.1%
$\alpha - \epsilon$	6,014,294	64.6%	7,865,072	67.9%	4,357,620	62.7%
$\alpha + \epsilon$	1,435,576	15.4%	1,592,655	13.7%	1,163,296	16.7%
15	1,377,375	14.8%	1,533,610	13.2%	1,115,559	16.0%
20	1,152,780	12.4%	1,291,113	11.1%	928,855	13.4%
25	905,201	9.7%	1,000,264	8.6%	732,230	10.5%
50	446,757	4.8%	481,737	4.2%	365,118	5.3%
100	235,132	2.5%	251,999	2.2%	189,655	2.7%
1000	22,873	0.2%	24,070	0.2%	16,467	0.2%

Table 4: Chinese–English: Distinct phrase pairs by pruning threshold

threshold	best		UN-v2	
none	18,858,589	100%	20,228,273	100%
14	7,666,063	40.7%	13,276,885	65.6%
16	4,280,845	22.7%	7,691,660	38.0%
18	4,084,167	21.7%	7,434,939	36.8%
20	3,887,397	20.6%	7,145,827	35.3%
25	3,403,674	18.0%	6,316,795	31.2%
	also pruning $C(\tilde{s}, \tilde{t}) = 1$			
$14'$	4,477,920	23.7%	7,917,062	39.1%

that an artificial separation of 1 BLEU point has been introduced into these graphs to separate them. Without this, they lie on top of each other and hide the essential point. In compensation, the scale for the BLEU co-ordinate has been removed.

These results are summarized in the following subsections.

4.1 BLEU as a function of threshold

In tables 5 and 6, the largest BLEU score for each set of runs has been marked in bold font. In addition, to highlight that there are many near ties for largest BLEU, all BLEU scores that are within 0.1 of the best are also marked in bold.

When this is done it becomes clear that pruning at a level of 20 for the WMT06 runs would not reduce BLEU in most cases and in many cases would actually increase it. A pruning threshold of 20 corresponds to discarding roughly 90% of the phrasetable.

For the Chinese–English large corpus runs, a level of 16 seems to be about the best with a small increase in BLEU and a $60\% - 70\%$ reduction in the size of the phrasetable.

4.2 BLEU as a function of depth of pruning

Another view of this can be taken from Tables 5 and 6. The fraction of the phrasetable retained is a more or less simple function of pruning threshold as shown in Tables 3 and 4. By including the percentages in Tables 5 and 6, we can see that BLEU goes up as the fraction approaches between 20% and 30%.

This seems to be a relatively stable observation across the experiments. It is also easily explained by its strong relationship to pruning threshold.

4.3 Large corpora

Table 6 shows that this is not just a small corpus phenomenon. There is a sizeable benefit both in phrasetable reduction and a modest improvement to BLEU even in this case.

4.4 Is this just the same as phrasetable smoothing?

One question that occurred early on was whether this improvement in BLEU is somehow related to the improvement in BLEU that occurs with phrasetable smoothing.

It appears that the answer is, in the main, yes, although there is definitely something else going on. It is true that the benefit in terms of BLEU is lessened for better types of phrasetable smoothing but the benefit in terms of the reduction in bulk holds. It is reassuring to see that no harm to BLEU is done by removing even 80% of the phrasetable.

4.5 Comment about $C(\tilde{s}, \tilde{t}) = 1$

Another question that came up is the role of phrase pairs that occur only once: $C(\tilde{s}, \tilde{t}) = 1$. In particular as discussed above, the most significant of these are the 1-1-1 phrase pairs whose components also only occur once: $C(\tilde{s}) = 1$, and $C(\tilde{t}) = 1$. These phrase pairs are amazingly frequent in the phrasetables and are pruned in all of the experiments except when pruning threshold is equal to 14.

The Chinese–English large corpus experiments give us a good opportunity to show that significance level seems to be more an issue than the case that $C(\tilde{s}, \tilde{t}) = 1$.

Note that we could have kept the phrase pairs whose marginal counts were greater than one but most of these are of lower significance and likely are pruned already by the threshold. The given configuration was considered the most likely to yield a benefit and its poor performance led to the whole idea being put aside.

5 Conclusions and Continuing Work

To sum up, the main conclusions are five in number:

1. Phrasetables produced by the standard Diag-And method (Koehn et al., 2003) can be aggressively pruned using significance pruning without worsening BLEU.

2. If phrasetable smoothing is not done, the BLEU score will improve under aggressive significance pruning.

3. If phrasetable smoothing is done, the improvement is small or negligible but there is still no loss on aggressive pruning.

4. The preservation of BLEU score in the presence of large-scale pruning is a strong effect in small and moderate size phrasetables, but occurs also in much larger phrasetables.

5. In larger phrasetables based on larger corpora, the percentage of the table that can be discarded appears to decrease. This is plausible since a similar effect (a decrease in the benefit of smoothing) has been noted with phrasetable smoothing (Foster et al., 2006). Together these results suggest that, for these corpus sizes, the increase in the number of strongly supported phrase pairs is greater than the increase in the number of poorly supported pairs, which agrees with intuition.

Although there may be other approaches to pruning that achieve a similar effect, the use of Fisher's exact test is mathematically and conceptually one of the simplest since it asks a question separately for each phrase pair: "Considering this phase pair in isolation of any other analysis on the corpus, could it have occurred plausibly by purely random processes inherent in the corpus construction?" If the answer is "Yes", then it is hard to argue that the phrase pair is an association of general applicability from the evidence in this corpus alone.

Note that the removal of 1-count phrase pairs is subsumed by significance pruning with a threshold greater than α and many of the other simple approaches (from an implementation point of view) are more difficult to justify as simply as the above significance test. Nonetheless, there remains work to do in determining if computationally simpler approaches do as well. Moore's work suggests that log-likelihood-ratio would be a cheaper and accurate enough alternative, for example.

We will now return to the interaction of the selection in our beam search of the top 30 candidates based on forward conditional probabilities. This will affect our results but most likely in the following manner:

1. For very small thresholds, the beam will become much wider and the search will take much longer. In order to allow the experiments to complete in a reasonable time, other means will need to be employed to reduce the choices. This reduction will also interact with the significance pruning but in a less understandable manner.

2. For large thresholds, there will not be 30

choices and so there will be no effect.

3. For intermediate thresholds, the extra pruning might reduce BLEU score but by a small amount because most of the best choices are included in the search.

Using thresholds that remove most of the phrasetable would no doubt qualify as large thresholds so the question is addressing the true shape of the curve for smaller thresholds and not at the expected operating levels. Nonetheless, this is a subject for further study, especially as we consider alternatives to our "filter 30" approach for managing beam width.

There are a number of important ways that this work can and will be continued. The code base for taking a list of n, m-grams and computing the required frequencies for signifance evaluation can be applied to related problems. For example, skip-n-grams (n-grams that allow for gaps of fixed or variable size) may be studied better using this approach leading to insight about methods that weakly approximate patterns.

The original goal of this work was to better understand the character of phrasetables, and it remains a useful diagnostic technique. It will hopefully lead to more understanding of what it takes to make a good phrasetable especially for languages that require morphological analysis or segmentation to produce good tables using standard methods.

The negative-log-p-value promises to be a useful feature and we are currently evaluating its merits.

6 Acknowledgement

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA)."

References

Alan Agresti. 1996. An Introduction to Categorical Data Analysis. Wiley.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

Philipp Koehn 2003. Europarl: A Multilingual Corpus for Evaluation of Machine Translation. Unpublished draft. see `http://www.iccs.inf.ed.ac.uk/~pkoehn/publications/europarl.pdf`

George Foster, Roland Kuhn, and Howard Johnson. 2006. Phrasetable Smoothing for Statistical Machine Translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1995*, pages 181–184, Detroit, Michigan. IEEE.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In Eduard Hovy, editor, *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133, Edmonton, Alberta, Canada, May. NAACL.

Robert C. Moore. 2004. On Log-Likelihood-Ratios and the Significance of Rare Events. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

NIST. 2006. NIST MT Benchmark Test. see `http://www.nist.gov/speech/tests/mt/`

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics(ACL)*, Sapporo, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of Machine Translation. Technical Report RC22176, IBM, September.

NAACL Workshop on Statistical Machine Translation. 2006. see `http://www.statmt.org/wmt06/`

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP) 2002*, Denver, Colorado, September.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of Human Language Technology Conference / North American Chapter of the ACL*, Boston, May.

Table 5: WMT06 Results: BLEU by type of smoothing and pruning threshold

threshold	phrasetable %	fr ⟶ en	es ⟶ en	de ⟶ en	en ⟶ fr	en ⟶ es	en ⟶ de
relative frequency: no smoothing							
none	100%	25.39	27.26	20.74	27.29	27.17	14.71
10	84–88%	25.97	27.81	21.08	27.82	27.71	15.09
$\alpha - \epsilon$	63–68%	26.32	28.00	**21.27**	28.11	**28.09**	**15.19**
$\alpha + \epsilon$	14–17%	26.34	28.27	21.22	28.16	**28.08**	**15.24**
15	13–15%	26.36	**28.50**	21.14	28.20	**28.18**	**15.29**
20	11–13%	**26.51**	**28.45**	**21.36**	**28.28**	28.06	**15.28**
25	8–10%	**26.50**	28.38	21.28	**28.32**	27.97	**15.25**
50	4–5%	26.26	27.88	20.87	28.05	27.90	15.08
100	2%	25.66	27.07	20.07	27.38	27.11	14.66
1000	0.2%	20.49	21.66	15.23	22.51	22.31	11.36
Good-Turing							
none	100%	25.96	28.14	21.17	27.84	27.95	15.13
10	84–88%	26.33	28.33	21.38	28.18	28.27	**15.22**
$\alpha - \epsilon$	63–68%	26.54	**28.63**	**21.50**	**28.36**	**28.39**	**15.31**
$\alpha + \epsilon$	14–17%	26.24	28.49	21.15	28.22	28.16	**15.28**
15	13–15%	26.48	28.03	21.21	**28.27**	28.21	**15.31**
20	11–13%	**26.65**	28.45	**21.41**	**28.36**	28.14	**15.25**
25	8–10%	26.54	**28.56**	21.31	**28.35**	28.04	**15.28**
50	4–5%	26.26	27.78	20.94	28.07	27.95	15.08
100	2%	25.70	27.07	20.12	27.41	27.13	14.66
1000	0.2%	20.49	21.66	15.52	22.53	22.31	11.37
Kneser-Ney (3 parameter)							
none	100%	**26.89**	**28.70**	**21.78**	**28.64**	**28.71**	**15.50**
10	84–88%	**26.79**	**28.78**	**21.71**	**28.63**	28.41	15.35
15	13–15%	26.49	**28.69**	21.34	**28.60**	28.57	**15.52**
20	11–13%	26.73	28.67	21.54	**28.56**	28.44	15.41
25	8–10%	**26.84**	**28.70**	21.29	**28.54**	28.21	**15.42**
50	4–5%	26.44	28.16	20.93	28.17	28.05	15.17
100	2%	25.72	27.27	20.11	27.50	27.26	14.58
1000	0.2%	20.48	21.70	15.28	22.58	22.36	11.33
Zens-Ney							
none	100%	**26.87**	**29.07**	21.55	**28.75**	**28.54**	**15.50**
10	84–88%	26.81	**29.00**	**21.65**	**28.72**	**28.52**	**15.54**
15	13–15%	**26.92**	28.67	**21.74**	**28.79**	28.32	**15.44**
20	11–13%	**26.93**	28.47	**21.72**	**28.69**	28.42	**15.45**
25	8–10%	**26.85**	28.79	21.58	28.59	28.27	15.37
50	4–5%	26.51	27.96	20.96	28.30	27.96	15.27
100	2%	25.82	27.34	20.02	27.57	27.30	14.51
1000	0.2%	20.50	21.76	15.46	22.68	22.33	11.56

Table 6: Chinese Results: BLEU by pruning threshold

threshold	phrasetable %	nist04	nist05	nist06-GALE	nist06-NIST
Zens-Ney Smoothing applied to all phrasetables					
none	100%	32.14	30.69	**13.06**	27.97
14	40–65%	**32.66**	**31.14**	**13.11**	**28.35**
16	22–38%	**32.73**	30.97	**13.14**	28.00
18	21–36%	31.56	30.45	12.49	27.03
20	20–35%	32.00	30.73	12.50	27.33
25	18–31%	30.54	29.58	11.68	26.12
also pruning $C(\tilde{s}, \tilde{t}) = 1$					
$14'$	23–39%	32.08	30.99	12.75	27.66

Hierarchical Phrase-Based Translation with Suffix Arrays

Adam Lopez
Computer Science Department
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742 USA
alopez@cs.umd.edu

Abstract

A major engineering challenge in statistical machine translation systems is the efficient representation of extremely large translation rulesets. In phrase-based models, this problem can be addressed by storing the training data in memory and using a suffix array as an efficient index to quickly lookup and extract rules on the fly. *Hierarchical* phrase-based translation introduces the added wrinkle of source phrases with gaps. Lookup algorithms used for contiguous phrases no longer apply and the best approximate pattern matching algorithms are much too slow, taking several minutes per sentence. We describe new lookup algorithms for hierarchical phrase-based translation that reduce the empirical computation time by nearly two orders of magnitude, making on-the-fly lookup feasible for source phrases with gaps.

1 Introduction

Current statistical machine translation systems rely on very large rule sets. In phrase-based systems, rules are extracted from parallel corpora containing tens or hundreds of millions of words. This can result in millions of rules using even the most conservative extraction heuristics. Efficient algorithms for rule storage and access are necessary for practical decoding algorithms. They are crucial to keeping up with the ever-increasing size of parallel corpora, as well as the introduction of new data sources such as web-mined and comparable corpora.

Until recently, most approaches to this problem involved substantial tradeoffs. The common practice of test set filtering renders systems impractical for all but batch processing. Tight restrictions on phrase length curtail the power of phrase-based models. However, some promising engineering solutions are emerging. Zens and Ney (2007) use a disk-based prefix tree, enabling efficient access to phrase tables much too large to fit in main memory. An alternative approach introduced independently by both Callison-Burch et al. (2005) and Zhang and Vogel (2005) is to store the training data itself in memory, and use a *suffix array* as an efficient index to look up, extract, and score phrase pairs on the fly. We believe that the latter approach has several important applications (§7).

So far, these techniques have focused on phrase-based models using *contiguous* phrases (Koehn et al., 2003; Och and Ney, 2004). Some recent models permit *discontiguous* phrases (Chiang, 2007; Quirk et al., 2005; Simard et al., 2005). Of particular interest to us is the hierarchical phrase-based model of Chiang (2007), which has been shown to be superior to phrase-based models. The ruleset extracted by this model is a superset of the ruleset in an equivalent phrase-based model, and it is an order of magnitude larger. This makes efficient rule representation even more critical. We tackle the problem using the online rule extraction method of Callison-Burch et al. (2005) and Zhang and Vogel (2005).

The problem statement for our work is: *Given an input sentence, efficiently find all hierarchical phrase-based translation rules for that sentence in the training corpus.*

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 976–985, Prague, June 2007. ©2007 Association for Computational Linguistics

We first review suffix arrays (§2) and hierarchical phrase-based translation (§3). We show that the obvious approach using state-of-the-art pattern matching algorithms is hopelessly inefficient (§4). We then describe a series of algorithms to address this inefficiency (§5). Our algorithms reduce computation time by two orders of magnitude, making the approach feasible (§6). We close with a discussion that describes several applications of our work (§7).

2 Suffix Arrays

A suffix array is a data structure representing all suffixes of a corpus in lexicographical order (Manber and Myers, 1993). Formally, for a text T, the ith suffix of T is the substring of the text beginning at position i and continuing to the end of T. This suffix can be uniquely identified by the index i of its first word. The suffix array SA_T of T is a permutation of $[1, |T|]$ arranged by the lexicographical order of the corresponding suffixes. This representation enables fast lookup of any contiguous substring using binary search. Specifically, all occurrences of a length-m substring can be found in $O(m + \log |T|)$ time (Manber and Myers, 1993). [1]

Callison-Burch et al. (2005) and Zhang and Vogel (2005) use suffix arrays as follows.

1. Load the source training text F, the suffix array SA_F, the target training text E, and the alignment A into memory.

2. For each input sentence, look up each substring (phrase) $\bar{f}$ of the sentence in the suffix array.

3. For each instance of $\bar{f}$ found in F, find its aligned phrase $\bar{e}$ using the phrase extraction method of Koehn et al. (2003).

4. Compute the relative frequency score $p(\bar{e}|\bar{f})$ of each pair using the count of the extracted pair and the marginal count of $\bar{f}$.

5. Compute the lexical weighting score of the phrase pair using the alignment that gives the best score.

6. Use the scored rules to translate the input sentence with a standard decoding algorithm.

A difficulty with this approach is step 3, which can be quite slow. Its complexity is linear in the number of occurrences of the source phrase $\bar{f}$. Both Callison-Burch et al. (2005) and Zhang and Vogel (2005) solve this with sampling. If a source phrase appears more than k times, they sample only k occurrences for rule extraction. Both papers report that translation performance is nearly identical to extracting all possible phrases when $k = 100$. [2]

3 Hierarchical Phrase-Based Translation

We consider the hierarchical translation model of Chiang (2007). Formally, this model is a synchronous context-free grammar. The lexicalized translation rules of the grammar may contain a single nonterminal symbol, denoted X. We will use a, b, c and d to denote terminal symbols, and u, v, and w to denote (possibly empty) sequences of these terminals. We will additionally use α and β to denote (possibly empty) sequences containing both terminals and nonterminals.

A translation rule is written $X \rightarrow \alpha/\beta$. This rule states that a span of the input matching α is replaced by β in translation. We require that α and β contain an equal number (possibly zero) of *coindexed* nonterminals. An example rule with coindexes is $X \rightarrow uX_{\boxed{1}}vX_{\boxed{2}}w/u'X_{\boxed{2}}v'X_{\boxed{1}}w'$. When discussing only the source side of such rules, we will leave out the coindexes. For instance, the source side of the above rule will be written $uXvXw$. [3]

For the purposes of this paper, we adhere to the restrictions described by Chiang (2007) for rules extracted from the training data.

- Rules can contain at most two nonterminals.
- Rules can contain at most five terminals.
- Rules can span at most ten words.

[1] Abouelhoda et al. (2004) show that lookup can be done in optimal $O(m)$ time using some auxiliaray data structures. For our purposes $O(m + \log |T|)$ is practical, since for the 27M-word corpus used to carry out our experiments, $\log |T| \sim 25$.

[2] A sample size of 100 is actually quite small for many phrases, some of which occur tens or hundreds of thousands of times. It is perhaps surprising that such a small sample size works as well as the full data. However, recent work by Och (2005) and Federico and Bertoldi (2006) has shown that the statistics used by phrase-based systems are not very precise.

[3] In the canonical representation of the grammar, source-side coindexes are always in sorted order, making them unambiguous.

- Nonterminals must span at least two words.
- Adjacent nonterminals are disallowed in the source side of a rule.

Expressed more economically, we say that our goal is to search for source phrases in the form u, uXv, or $uXvXw$, where $1 \leq |uvw| \leq 5$, and $|v| > 0$ in the final case. Note that the model also allows rules in the form Xu, uX, XuX, $XuXv$, and $uXvX$. However, these rules are lexically identical to other rules, and thus will match the same locations in the source text.

4 The Collocation Problem

On-the-fly lookup using suffix arrays involves an added complication when the rules are in form uXv or $uXvXw$. Binary search enables fast lookup of contiguous substrings. However, it cannot be used for discontiguous substrings. Consider the rule $aXbXc$. If we search for this rule in the following logical suffix array fragment, we will find the bold-faced matches.

...

a	c	a	c	**b**	a	d	**c**	a	d	...
a	c	a	d	b	a	a	d	b	d	...
a	d	d	**b**	a	a	d	a	b	**c**	...
a	d	d	b	d	a	a	b	b	a	...
a	d	d	**b**	d	d	**c**	a	a	a	...

...

Even though these suffixes are in lexicographical order, matching suffixes are interspersed with non-matching suffixes. We will need another algorithm to find the source rules containing at least one X surrounded by nonempty sequences of terminal symbols.

4.1 Baseline Approach

In the pattern-matching literature, words spanned by the nonterminal symbols of Chiang's grammar are called *don't cares* and a nonterminal symbol in a query pattern that matches a sequence of don't cares is called a *variable length gap*. The search problem for patterns containing these gaps is a variant of *approximate pattern matching*, which has received substantial attention (Navarro, 2001). The best algorithm for pattern matching with variable-length gaps in a suffix array is a recent algorithm by Rahman et al. (2006). It works on a pattern $w_1Xw_2X...w_I$ consisting of I contiguous substrings $w_1, w_2, ...w_I$, each separated by a gap. The algorithm is straightforward. After identifying all n_i occurrences of each w_i in $O(|w_i| + \log|T|)$ time, collocations that meet the gap constraints are computed using an efficient data structure called a *stratified tree* (van Emde Boas et al., 1977). [4] Although we refer the reader to the source text for a full description of this data structure, its salient characteristic is that it implements priority queue operations *insert* and *next-element* in $O(\log\log|T|)$ time. Therefore, the total running time for an algorithm to find all contiguous subpatterns and compute their collocations is $O(\sum_{i=1}^{I}[|w_i| + log|T| + n_i \log\log|T|])$.

We can improve on the algorithm of Rahman et al. (2006) using a variation on the idea of hashing. We exploit the fact that our large text is actually a collection of relatively short sentences, and that collocated patterns must occur in the same sentence in order to be considered a rule. Therefore, we can use the sentence id of each subpattern occurrence as a kind of hash key. We create a hash table whose size is exactly the number of sentences in our training corpus. Each location of the partially matched pattern $w_1X...Xw_i$ is inserted into the hash bucket with the matching sentence id. To find collocated patterns w_{i+1}, we probe the hash table with each of the n_{i+1} locations for that subpattern. When a match is found, we compare the element with all elements in the bucket to see if it is within the window imposed by the phrase length constraints. Theoretically, the worst case for this algorithm occurs when all elements of both sets resolve to the same hash bucket, and we must compare all elements of one set with all elements of the other set. This leads to a worst case complexity of $O(\sum_{i=1}^{I}[|w_i| + log|T|] + \prod_{i=1}^{I} n_i)$. However, for real language data the performance for sets of any significant size will be $O(\sum_{i=1}^{I}[|w_i| + log|T| + n_i])$, since most patterns will occur once in any given sentence.

4.2 Analysis

It is instructive to compare this with the complexity for contiguous phrases. In that case, total lookup time is $O(|w| + log|T|)$ for a contiguous pattern w.

[4] Often known in the literature as a *van Emde Boas tree* or *van Emde Boas priority queue*.

The crucial difference between the contiguous and discontiguous case is the added term $\sum_{i=1}^{I} n_i$. For even moderately frequent subpatterns this term dominates complexity.

To make matters concrete, consider the training corpus used in our experiments (§6), which contains 27M source words. The three most frequent unigrams occur 1.48M, 1.16M and 688K times – the first two occur on average more than once per sentence. In the worst case, looking up a contiguous phrase containing any number and combination of these unigrams requires no more than 25 comparison operations. In contrast, the worst case scenario for a pattern with a single gap, bookended on either side by the most frequent word, requires over *two million* operations using our baseline algorithm and over *thirteen million* using the algorithm of Rahman et al. (2006). A single frequent word in an input sentence is enough to cause noticeable slowdowns, since it can appear in up to 530 hierarchical rules.

To analyze the cost empirically, we ran our baseline algorithm on the first 50 sentences of the NIST Chinese-English 2003 test set and measured the CPU time taken to compute collocations. We found that, on average, it took 2241.25 seconds (∼37 minutes) *per sentence* just to compute all of the needed collocations. By comparison, decoding time per sentence is roughly 10 seconds with moderately aggressive pruning, using the Python implementation of Chiang (2007).

5 Solving the Collocation Problem

Clearly, looking up patterns in this way is not practical. To analyze the problem, we measured the amount of CPU time per computation. Cumulative lookup time was dominated by a very small fraction of the computations (Fig. 1). As expected, further analysis showed that these expensive computations all involved one or more very frequent subpatterns. In the worst cases a single collocation took several seconds to compute. However, there is a silver lining. Patterns follow a Zipf distribution, so the number of pattern *types* that cause the problem is actually quite small. The vast majority of patterns are rare. Therefore, our solution focuses on computations where one or more of the component patterns is frequent. Assume that we are computing a collocation of pattern $w_1X...Xw_i$ and pattern w_{i+1}, and we know all locations of each. There are three cases.

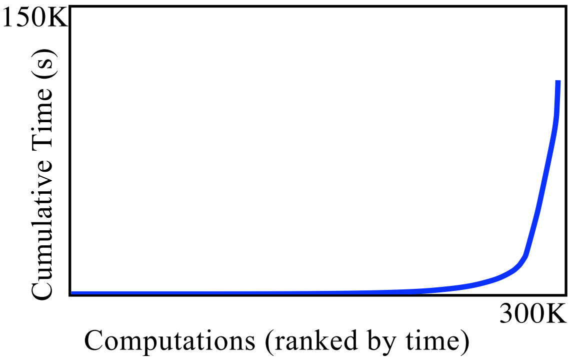


Figure 1: Ranked computations vs. cumulative time. A small fraction of all computations account for most of the computational time.

- If both patterns are frequent, we resort to a precomputed intersection (§5.1). We were not aware of any algorithms to substantially improve the efficiency of this computation when it is requested on the fly, but precomputation can be done in a single pass over the text at decoder startup.
- If one pattern is frequent and the other is rare, we use an algorithm whose complexity is dependent mainly on the frequency of the rare pattern (§5.2). It can also be used for pairs of rare patterns when one pattern is much rarer than the other.
- If both patterns are rare, no special algorithms are needed. Any linear algorithm will suffice. However, for reasons described in §5.3, our other collocation algorithms depend on sorted sets, so we use a merge algorithm.

Finally, in order to cut down on the number of unnecessary computations, we use an efficient method to enumerate the phrases to lookup (§5.4). This method also forms the basis of various caching strategies for additional speedups. We analyze the memory use of our algorithms in §5.5.

5.1 Precomputation

Precomputation of the most expensive collocations can be done in a single pass over the text. As input, our algorithm requires the identities of the k

most frequent contiguous patterns. [5] It then iterates over the corpus. Whenever a pattern from the list is seen, we push a tuple consisting of its identity and current location onto a queue. Whenever the oldest item on the queue falls outside the maximum phrase length window with respect to the current position, we compute that item's collocation with all succeeding patterns (subject to pattern length constraints) and pop it from the queue. We repeat this step for every item that falls outside the window. At the end of each sentence, we compute collocations for any remaining items in the queue and then empty it.

Our precomputation includes the most frequent n-gram subpatterns. Most of these are unigrams, but in our experiments we found 5-grams among the 1000 most frequent patterns. We precompute the locations of source phrase uXv for any pair u and v that both appear on this list. There is also a small number of patterns uXv that are very frequent. We cannot easily obtain a list of these in advance, but we observe that they always consist of a pair u and v of patterns from near the top of the frequency list. Therefore we also precompute the locations $uXvXw$ of patterns in which both u and v are among these *super-frequent* patterns (all unigrams), treating this as the collocation of the frequent pattern uXv and frequent pattern w. We also compute the analagous case for u and vXw.

5.2 Fast Intersection

For collocations of frequent and rare patterns, we use a fast set intersection method for sorted sets called double binary search (Baeza-Yates, 2004). [6] It is based on the intuition that if one set in a pair of sorted sets is much smaller than the other, then we can compute their intersection efficiently by performing a binary search in the larger *data set* D for each element of the smaller *query set* Q.

Double binary search takes this idea a step further. It performs a binary search in D for the median element of Q. Whether or not the element is found, the search divides both sets into two pairs of smaller sets that can be processed recursively. Detailed analysis and empirical results on an information retrieval task are reported in Baeza-Yates (2004) and Baeza-Yates and Salinger (2005). If $|Q| \log |D| < |D|$ then the performance is guaranteed to be sublinear. In practice it is often sublinear even if $|Q| \log |D|$ is somewhat larger than $|D|$. In our implementation we simply check for the condition $\lambda|Q| \log |D| < |D|$ to decide whether we should use double binary search or the merge algorithm. This check is applied in the recursive cases as well as for the initial inputs. The variable λ can be adjusted for performance. We determined experimentally that a good value for this parameter is 0.3.

5.3 Obtaining Sorted Sets

Double binary search requires that its input sets be in sorted order. However, the suffix array returns matchings in lexicographical order, not numeric order. The algorithm of Rahman et al. (2006) deals with this problem by inserting the unordered items into a stratified tree. This requires $O(n \log\log |T|)$ time for n items. If we used the same strategy, our algorithm would no longer be sublinear.

An alternative is to precompute all n-gram occurrences in order and store them in an inverted index. This can be done in one pass over the data. [7] This approach requires a separate inverted index for each n, up to the maximum n used by the model. The memory cost is one length-$|T|$ array per index.

In order to avoid the full $n|T|$ cost in memory, our implementation uses a mixed strategy. We keep a precomputed inverted index only for unigrams. For bigrams and larger n-grams, we generate the index on the fly using stratified trees. This results in a superlinear algorithm for intersection. However, we can exploit the fact that we must compute collocations multiple times for each input n-gram by caching the sorted set after we create it (The caching strategy is described in §5.4). Subsequent computations involving this n-gram can then be done in linear or sublinear time. Therefore, the cost of building the inverted index on the fly is amortized over a large number of computations.

[5] These can be identified using a single traversal over a *longest common prefix (LCP) array*, an auxiliary data structure of the suffix array, described by Manber and Myers (1993). Since we don't need the LCP array at runtime, we chose to do this computation once offline.

[6] Minor modifications are required since we are computing collocation rather than intersection. Due to space constraints, details and proof of correctness are available in Lopez (2007a).

[7] We combine this step with the other precomputations that require a pass over the data, thereby removing a redundant $O(|T|)$ term from the startup cost.

5.4 Efficient Enumeration

A major difference between contiguous phrase-based models and hierarchical phrase-based models is the number of rules that potentially apply to an input sentence. To make this concrete, on our data, with an average of 29 words per sentence, there were on average 133 contiguous phrases of length 5 or less that applied. By comparison, there were on average 7557 hierarchical phrases containing up to 5 words. These patterns are obviously highly overlapping and we employ an algorithm to exploit this fact. We first describe a baseline algorithm used for contiguous phrases (§5.4.1). We then introduce some improvements (§5.4.2) and describe a data structure used by the algorithm (§5.4.3). Finally, we discuss some special cases for discontiguous phrases (§5.4.4).

5.4.1 The Zhang-Vogel Algorithm

Zhang and Vogel (2005) present a clever algorithm for contiguous phrase searches in a suffix array. It exploits the fact that for each m-length source phrase that we want to look up, we will also want to look up its $(m-1)$-length prefix. They observe that the region of the suffix array containing all suffixes prefixed by ua is a subset of the region containing the suffixes prefixed by u. Therefore, if we enumerate the phrases of our sentence in such a way that we always search for u before searching for ua, we can restrict the binary search for ua to the range containing the suffixes prefixed by u. If the search for u fails, we do not need to search for ua at all. They show that this approach leads to some time savings for phrase search, although the gains are relatively modest since the search for contiguous phrases is not very expensive to begin with. However, the potential savings in the discontiguous case are much greater.

5.4.2 Improvements and Extensions

We can improve on the Zhang-Vogel algorithm. An m-length contiguous phrase aub depends not only on the existence of its prefix au, but also on the existence of its suffix ub. In the contiguous case, we cannot use this information to restrict the starting range of the binary search, but we can check for the existence of ub to decide whether we even need to search for aub at all. This can help us avoid searches that are guaranteed to be fruitless.

Now consider the discontiguous case. As in the analogous contiguous case, a phrase $a\alpha b$ will only exist in the text if its *maximal prefix* $a\alpha$ and *maximal suffix* αb both exist in the corpus and overlap at specific positions. [8] Searching for $a\alpha b$ is potentially very expensive, so we put all available information to work. Before searching, we require that both $a\alpha$ and αb exist. Additionally, we compute the location of $a\alpha b$ using the locations of both maximal subphrases. To see why the latter optimization is useful, consider a phrase $abXcd$. In our baseline algorithm, we would search for ab and cd, and then perform a computation to see whether these subphrases were collocated within an elastic window. However, if we instead use $abXc$ and $bXcd$ as the basis of the computation, we gain two advantages. First, the number elements of each set is likely to be smaller then in the former case. Second, the computation becomes simpler, because we now only need to check to see whether the patterns exactly overlap with a starting offset of one, rather than checking within a window of locations.

We can improve efficiency even further if we consider cases where the same substring occurs more than once within the same sentence, or even in multiple sentences. If the computation required to look up a phrase is expensive, we would like to perform the lookup only once. This requires some mechanism for caching. Depending on the situation, we might want to cache only certain subsets of phrases, based on their frequency or difficulty to compute. We would also like the flexibility to combine on-the-fly lookups with a partially precomputed phrase table, as in the online/offline mixture of Zhang and Vogel (2005).

We need a data structure that provides this flexibility, in addition to providing fast access to both the maximal prefix and maximal suffix of any phrase that we might consider.

5.4.3 Prefix Trees and Suffix Links

Our search optimizations are easily captured in a *prefix tree* data structure augmented with *suffix links*. Formally, a prefix tree is an unminimized deterministic finite-state automaton that recognizes all of the patterns in some set. Each node in the tree repre-

[8] Except when $\alpha = X$, in which case a and b must be collocated within a window defined by the phrase length constraints.

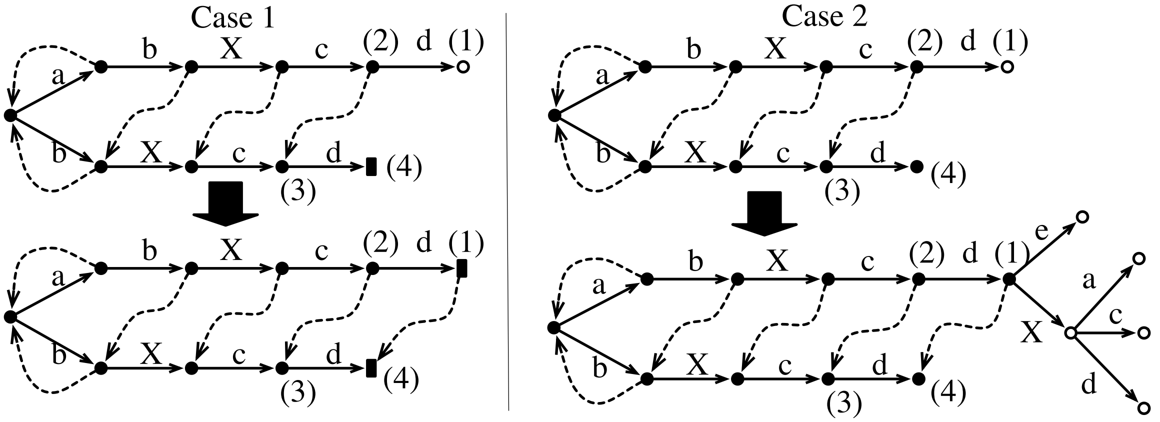


Figure 2: Illustration of prefix tree construction showing a partial prefix tree, including suffix links. Suppose we are interested in pattern $abXcd$, represented by node (1). Its prefix is represented by node (2), and node (2)'s suffix is represented by node (3). Therefore, node (1)'s suffix is represented by the node pointed to by the d-edge from node (3), which is node (4). There are two cases. In case 1, node (4) is inactive, so we can mark node (1) inactive and stop. In case 2, node (4) is active, so we compute the collocation of $abXc$ and $bXcd$ with information stored at nodes (2) and (4), using either a precomputed intersection, double binary search, or merge, depending on the size of the sets. If the result is empty, we mark the node inactive. Otherwise, we store the results at node (1) and add its successor patterns to the frontier for the next iteration. This includes all patterns containing exactly one more terminal symbol than the current pattern.

sents the prefix of a unique pattern from the set that is specified by the concatenation of the edge labels along the path from the root to that node. A suffix link is a pointer from a node representing path $a\alpha$ to the node representing path α. We will use this data structure to record the set of patterns that we have searched for and to cache information for those that were found successfully.

Our algorithm generates the tree breadth-search along a *frontier*. In the mth iteration we only search for patterns containing m terminal symbols. Regardless of whether we find a particular pattern, we create a node for it in the tree. If the pattern was found in the corpus, its node is marked *active*. Otherwise, it is marked *inactive*. For found patterns, we store either the endpoints of the suffix array range containing the phrase (if it is contiguous), or the list of locations at which the phrase is found (if it is discontiguous). We can also store the extracted rules. [9] Whenever a pattern is successfully found, we add all patterns with $m+1$ terminals that are prefixed by it to the frontier for processing in the next iteration.

To search for a pattern, we use location information from its parent node, which represents its maximal prefix. Assuming that the node represents phrase αb, we find the node representing its maximal suffix by following the b-edge from the node pointed to by its parent node's suffix link. If the node pointed to by this suffix link is inactive, we can mark the node inactive without running a search. When a node is marked inactive, we discontinue search for phrases that are prefixed by the path it represents. The algorithm is illustrated in Figure 2.

5.4.4 Special Cases for Phrases with Gaps

A few subtleties arise in the extraction of hierarchical patterns. Gaps are allowed to occur at the beginning or end of a phrase. For instance, we may have a source phrase Xu or uX or even XuX. Although each of these phrases requires its own path in the prefix tree, they are lexically identical to phrase u. An analogous situation occurs with the patterns $XuXv$, $uXvX$, and uXv. There are two cases that we are concerned with.

The first case consists of all patterns prefixed with X. The paths to nodes representing these patterns

[9] Conveniently, the implementation of Chiang (2007) uses a prefix tree grammar encoding, as described in Klein and Manning (2001). Our implementation decorates this tree with additional information required by our algorithms.

will all contain the X-edge originating at the root node. All of these paths form the *shadow subtree*. Path construction in this subtree proceeds differently. Because they are lexically identical to their suffixes, they are automatically extended if their suffix paths are active, and they inherit location information of their suffixes.

The second case consists of all patterns suffixed with X. Whenever we successfully find a new pattern α, we automatically extend it with an X edge, provided that αX is allowed by the model constraints. The node pointed to by this edge inherits its location information from its parent node (representing the maximal prefix α).

Note that both special cases occur for patterns in the form XuX.

5.5 Memory Requirements

As shown in Callison-Burch et al. (2005), we must keep an array for the source text F, its suffix array, the target text E, and alignment A in memory. Assuming that A and E are roughly the size of F, the cost is $4|T|$. If we assume that all data use vocabularies that can be represented using 32-bit integers, then our 27M word corpus can easily be represented in around 500MB of memory. Adding the inverted index for unigrams increases this by 20%. The main additional cost in memory comes from the storage of the precomputed collocations. This is dependent both on the corpus size and the number of collocations that we choose to precompute. Using detailed timing data from our experiments we were able to simulate the memory-speed tradeoff (Fig. 3). If we include a trigram model trained on our bitext and the Chinese Gigaword corpus, the overall storage costs for our system are approximately 2GB.

6 Experiments

All of our experiments were performed on Chinese-English in the news domain. We used a large training set consisting of over 1 million sentences from various newswire corpora. This corpus is roughly the same as the one used for large-scale experiments by Chiang et al. (2005). To generate alignments, we used GIZA++ (Och and Ney, 2003). We symmetrized bidirectional alignments using the grow-diag-final heuristic (Koehn et al., 2003).

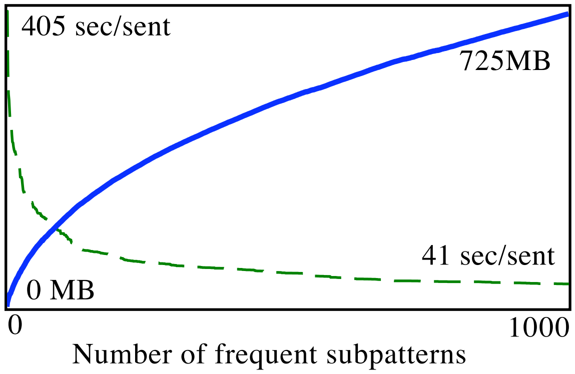


Figure 3: Effect of precomputation on memory use and processing time. Here we show only the memory requirements of the precomputed collocations.

We used the first 50 sentences of the NIST 2003 test set to compute timing results. All of our algorithms were implemented in Python 2.4. [10] Timing results are reported for machines with 8GB of memory and 4 3GHz Xeon processors running Red Hat linux 2.6.9. In order to understand the contributions of various improvements, we also ran the system with with various ablations. In the default setting, the prefix tree is constructed for each sentence to guide phrase lookup, and then discarded. To show the effect of caching we also ran the algorithm without discarding the prefix tree between sentences, resulting in full inter-sentence caching. The results are shown in Table 1. [11]

It is clear from the results that each of the optimizations is needed to sufficiently reduce lookup time to practical levels. Although this is still relatively slow, it is much closer to the decoding time of 10 seconds per sentence than the baseline.

[10] Python is an interpreted language and our implementations do not use any optimization features. It is therefore reasonable to think that a more efficient reimplementation would result in across-the-board speedups.

[11] The results shown here do not include the startup time required to load the data structures into memory. In our Python implementation this takes several minutes, which in principle should be amortized over the cost for each sentence. However, just as Zens and Ney (2007) do for phrase tables, we could compile our data structures into binary memory-mapped files, which can be read into memory in a matter of seconds. We are currently investigating this option in a C reimplementation.

Algorithms	Secs/Sent	Collocations
Baseline	2241.25	325548
Prefix Tree	1578.77	69994
Prefix Tree + precomputation	696.35	69994
Prefix Tree + double binary	405.02	69994
Prefix Tree + precomputation + double binary	40.77	69994
Prefix Tree with full caching + precomputation + double binary	30.70	67712

Table 1: Timing results and number of collocations computed for various combinations of algorithms. The runs using precomputation use the 1000 most frequent patterns.

7 Conclusions and Future Work

Our work solves a seemingly intractable problem and opens up a number of intriguing potential applications. Both Callison-Burch et al. (2005) and Zhang and Vogel (2005) use suffix arrays to relax the length constraints on phrase-based models. Our work enables this in hierarchical phrase-based models. However, we are interested in additional applications.

Recent work in discriminative learning for many natural language tasks, such as part-of-speech tagging and information extraction, has shown that feature engineering plays a critical role in these approaches. However, in machine translation most features can still be traced back to the IBM Models of 15 years ago (Lopez, 2007b). Recently, Lopez and Resnik (2006) showed that most of the features used in standard phrase-based models do not help very much. Our algorithms enable us to look up phrase pairs *in context*, which will allow us to compute interesting contextual features that can be used in discriminative learning algorithms to improve translation accuracy. Essentially, we can use the training data itself as an indirect representation of whatever features we might want to compute. This is not possible with table-based architectures.

Most of the data structures and algorithms discussed in this paper are widely used in bioinformatics, including suffix arrays, prefix trees, and suffix links (Gusfield, 1997). As discussed in §4.1, our problem is a variant of the approximate pattern matching problem. A major application of approximate pattern matching in bioinformatics is query processing in protein databases for purposes of sequencing, phylogeny, and motif identification.

Current MT models, including hierarchical models, translate by breaking the input sentence into small pieces and translating them largely independently. Using approximate pattern matching algorithms, we imagine that machine translation could be treated very much like search in a protein database. In this scenario, the goal is to select training sentences that match the input sentence as closely as possible, under some evaluation function that accounts for both matching and mismatched sequences, as well as possibly other data features. Once we have found the closest sentences we can translate the matched portions in their entirety, replacing mismatches with appropriate word, phrase, or hierarchical phrase translations as needed. This model would bring statistical machine translation closer to convergence with so-called *example-based translation*, following current trends (Marcu, 2001; Och, 2002). We intend to explore these ideas in future work.

Acknowledgements

I would like to thank Philip Resnik for encouragement, thoughtful discussions and wise counsel; David Chiang for providing the source code for his translation system; and Nitin Madnani, Smaranda Muresan and the anonymous reviewers for very helpful comments on earlier drafts of this paper. Any errors are my own. This research was supported in part by ONR MURI Contract FCPO.810548265 and the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the author and do not necessarily reflect the view of DARPA.

References

Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. 2004. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, Mar.

Ricardo Baeza-Yates and Alejandro Salinger. 2005. Experimental analysis of a fast intersection algorithm for sorted sequences. In M. Consens and G. Navarro, editors, *Proc. of SPIRE*, number 3772 in LNCS, pages 13–24, Berlin. Springer-Verlag.

Ricardo Baeza-Yates. 2004. A fast intersection algorithm for sorted sequences. In *Proc. of Combinatorial Pattern Matching*, number 3109 in LNCS, pages 400–408, Berlin. Springer-Verlag.

Chris Callison-Burch, Colin Bannard, and Josh Shroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proc. of ACL*, pages 255–262, Jun.

David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. 2005. The Hiero machine translation system: Extensions, evaluation, and analysis. In *Proc. of HLT-EMLP*, pages 779–786, Oct.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2). In press.

Marcello Federico and Nicola Bertoldi. 2006. How many bits are needed to store probabilities for phrase-based translation? In *Proc. of NAACL Workshop on Statistical Machine Translation*, pages 94–101, Jun.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press.

Dan Klein and Christopher D. Manning. 2001. Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proc. of ACL-EACL*, pages 330–337, Jul.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 127–133, May.

Adam Lopez and Philip Resnik. 2006. Word-based alignment, phrase-based translation: What's the link? In *Proc. of AMTA*, pages 90–99, Aug.

Adam Lopez. 2007a. Hierarchical phrase-based translation with suffix arrays. Technical Report 2007-26, University of Maryland Institute for Advanced Computer Studies, May.

Adam Lopez. 2007b. A survey of statistical machine translation. Technical Report 2006-47, University of Maryland Institute for Advanced Computer Studies, Apr.

Udi Manber and Gene Myers. 1993. Suffix arrays: A new method for on-line string searches. *SIAM Journal of Computing*, 22(5):935–948.

Daniel Marcu. 2001. Towards a unified approach to memory- and statistical-based machine translation. In *Proc. of ACL-EACL*, pages 378–385, Jul.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, Mar.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, Mar.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to machine translation. *Computational Linguistics*, 30(4):417–449, Jun.

Franz Josef Och. 2002. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.D. thesis, RWTH Aachen, Oct.

Franz Josef Och. 2005. Statistical machine translation: The fabulous present and future. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, Jun. Invited talk.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*, pages 271–279, Jun.

Mohammad Sohel Rahman, Costas S. Iliopoulos, Inbok Lee, Manal Mohamed, and William F. Smyth. 2006. Finding patterns with variable length gaps or don't cares. In *Proc. of COCOON*, Aug.

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. 2005. Translating with non-contiguous phrases. In *Proc. of HLT-EMNLP*, pages 755–762, Oct.

Peter van Emde Boas, R. Kaas, and E. Zijlstra. 1977. Design and implementation of an efficient priority queue. *Mathematical Systems Theory*, 10(2):99–127.

Richard Zens and Hermann Ney. 2007. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Proc. of HLT-NAACL*. To appear.

Ying Zhang and Stephan Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proc. of EAMT*, May.

An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems

Wolfgang Macherey
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA
wmach@google.com

Franz Josef Och
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA
och@google.com

Abstract

This paper presents an empirical study on how different selections of input translation systems affect translation quality in system combination. We give empirical evidence that the systems to be combined should be of similar quality and need to be almost uncorrelated in order to be beneficial for system combination. Experimental results are presented for composite translations computed from large numbers of different research systems as well as a set of translation systems derived from one of the best-ranked machine translation engines in the 2006 NIST machine translation evaluation.

1 Introduction

Computing consensus translations from the outputs of multiple machine translation engines has become a powerful means to improve translation quality in many machine translation tasks. Analogous to the ROVER approach in automatic speech recognition (Fiscus, 1997), a composite translation is computed by voting on the translation outputs of multiple machine translation systems. Depending on how the translations are combined and how the voting scheme is implemented, the composite translation may differ from any of the original hypotheses. While elementary approaches simply select for each sentence one of the original translations, more sophisticated methods allow for combining translations on a word or a phrase level.

Although system combination could be shown to result in substantial improvements in terms of translation quality (Matusov et al., 2006; Sim et al., 2007), not every possible ensemble of translation outputs has the potential to outperform the primary translation system. In fact, an adverse combination of translation systems may even deteriorate translation quality. This holds to a greater extent, when the ensemble of translation outputs contains a significant number of translations produced by low performing but highly correlated systems.

In this paper we present an empirical study on how different ensembles of translation outputs affect performance in system combination. In particular, we will address the following questions:

- *To what extent can translation quality benefit from combining systems developed by multiple research labs?*
  Despite an increasing number of translation engines, most state-of-the-art systems in statistical machine translation are nowadays based on implementations of the same techniques. For instance, word alignment models are often trained using the GIZA++ toolkit (Och and Ney, 2003); error minimizing training criteria such as the *Minimum Error Rate Training* (Och, 2003) are employed in order to learn feature function weights for log-linear models; and translation candidates are produced using phrase-based decoders (Koehn et al., 2003) in combination with n-gram language models (Brants et al., 2007).

  All these methods are established as *de facto* standards and form an integral part of most statistical machine translation systems. This, however, raises the question as to what extent translation quality can be expected to improve when similarly designed systems are combined.

- *How can a set of diverse translation systems be built from a single translation engine?*
  Without having access to different translation

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 986–995, Prague, June 2007. ©2007 Association for Computational Linguistics

engines, it is desirable to build a large number of diverse translation systems from a *single* translation engine that are useful in system combination. The mere use of N-best lists and word lattices is often not effective, because N-best candidates may be highly correlated, thus resulting in small diversity compared to the first best hypothesis. Therefore, we need a canonical way to build a large pool of diverse translation systems from a *single* translation engine.

- *How can an ensemble of translation outputs be selected from a large pool of translation systems?*
  Once a large pool of translation systems is available, we need an effective means to select a small ensemble of translation outputs for which the combined system outperforms the best individual system.

These questions will be investigated on the basis of three approaches to system combination: (i) an MBR-like candidate selection method based on *BLEU correlation matrices*, (ii) confusion networks built from word sausages, and (iii) a novel two-pass search algorithm that aims at finding consensus translations by reordering bags of words constituting the consensus hypothesis.

Experiments were performed on two Chinese-English text translation corpora under the conditions of the large data track as defined for the 2006 NIST machine translation evaluation (MT06). Results are reported for consensus translations built from system outputs provided by MT06 participants as well as systems derived from one of the best-ranked translation engines.

The remainder of this paper is organized as follows: in Section 2, we describe three combination methods for computing consensus translations. In Sections 3.1 and 3.2, we present experimental results on combining system outputs provided by MT06 participants. Section 3.3 shows how correlation among translation systems affects performance in system combination. In Section 3.4, we discuss how a single translation engine can be modified in order to produce a large number of diverse translation systems. First experimental results using a greedy search algorithm to select a small ensemble of translation outputs from a large pool of canonically built translation systems are reported. A summary presented in Section 4 concludes the paper.

2 Methods for System Combination

System combination in machine translation aims to build a composite translation from system outputs of multiple machine translation engines. Depending on how the systems are combined and which voting scheme is implemented, the consensus translation may differ from any of the original candidate translations. In this section, we discuss three approaches to system combination.

2.1 System Combination via Candidate Selection

The easiest and most straightforward approach to system combination simply returns one of the original candidate translations. Typically, this selection is made based on translation scores, confidence estimations, language and other models (Nomoto, 2004; Paul et al., 2005). For many machine translation systems, however, the scores are often not normalized or may even not be available, which makes it difficult to apply this technique. We therefore propose an alternative method based on "correlation matrices" computed from the BLEU performance measure (Papineni et al., 2001).

Let $\mathbf{e}_1, ..., \mathbf{e}_M$ denote the outputs of M translation systems, each given as a sequence of words in the target language. An element of the BLEU correlation matrix $\mathbf{B} = (b_{ij})$ is defined as the sentence-based BLEU score between a candidate translation $\mathbf{e}_i$ and a pseudo-reference translation $\mathbf{e}_j$ $(i, j = 1, ..., M)$:

$$b_{ij} = \mathrm{BP}(\mathbf{e}_i, \mathbf{e}_j) \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4} \log \rho_n(\mathbf{e}_i, \mathbf{e}_j)\right). \tag{1}$$

Here, BP denotes the brevity penalty factor with ρ_n designating the n-gram precisions.

Because the BLEU score is computed on a sentence rather than a corpus-level, n-gram precisions are capped by the maximum over $\frac{1}{2\cdot|\mathbf{e}_i|}$ and ρ_n in order to avoid singularities, where $|\mathbf{e}_i|$ is the length of the candidate translation [1].

Due to the following properties, $\mathbf{B}$ can be interpreted as a correlation matrix, although the term does not hold in a strict mathematical sense: (i) $b_{ij} \in [0, 1]$; (ii) $b_{ij} = 1.0 \iff \mathbf{e}_i = \mathbf{e}_j$; (iii) $b_{ij} = 0.0 \iff \mathbf{e}_i \cap \mathbf{e}_j = \varnothing$, i.e., b_{ij} is zero if and only if none of the words which constitute $\mathbf{e}_i$ can be found

[1] Note that for non-zero n-gram precisions, ρ_n is always larger than $\frac{1}{2\cdot|\mathbf{e}|}$.

in $\mathbf{e}_j$ and vice versa. The BLEU correlation matrix is in general, however, not symmetric, although in practice, $||b_{ij} - b_{ji}||$ is typically negligible.

Each translation system m is assigned to a *system prior weight* $\omega_m \in [0, 1]$, which reflects the performance of system m relatively to all other translation systems. If no prior knowledge is available, ω_m is set to $1/M$.

Now, let $\boldsymbol{\omega} = (\omega_1, ..., \omega_M)^\top$ denote a vector of system prior weights and let $\mathbf{b}_1, ..., \mathbf{b}_M$ denote the row vectors of the matrix $\mathbf{B}$. Then the translation system with the highest consensus is given by:

$$\begin{aligned} \mathbf{e}^* &= \mathbf{e}_{m^*} \quad \text{with} \\ m^* &= \underset{\mathbf{e}_m}{\operatorname{argmax}} \left\{ \boldsymbol{\omega}^\top \cdot \mathbf{b}_m \right\} \end{aligned} \tag{2}$$

The candidate selection rule in Eq. (2) has two useful properties:

- The selection does not depend on scored translation outputs; the mere target word sequence is sufficient. Hence, this technique is also applicable to rule-based translation systems [2].
- Using the components of the row-vector $\mathbf{b}_m$ as feature function values for the candidate translation $\mathbf{e}_m$ ($m = 1, ..., M$), the system prior weights $\boldsymbol{\omega}$ can easily be trained using the Minimum Error Rate Training described in (Och, 2003).

Note that the candidate selection rule in Eq. (2) is equivalent to re-ranking candidate translations according to the *Minimum Bayes Risk* (MBR) decision rule (Kumar and Byrne, 2004), provided that the system prior weights are used as estimations of the posterior probabilities $p(\mathbf{e}|\mathbf{f})$ for a source sentence $\mathbf{f}$. Due to the proximity of this method to the MBR selection rule, we call this combination scheme *MBR-like system combination*.

2.2 ROVER-Like Combination Schemes

ROVER-like combination schemes aim at computing a composite translation by voting on confusion networks that are built from translation outputs of multiple machine translation engines via an iterative application of alignments (Fiscus, 1997). To accomplish this, one of the original candidate translations, e.g. $\mathbf{e}_m$, is chosen as the primary translation hypothesis, while all other candidates $\mathbf{e}_n$ $(n \neq m)$ are aligned with the word sequence of the primary translation. To limit the costs when aligning a permutation of the primary translation, the alignment metric should allow for small shifts of contiguous word sequences in addition to the standard edit operations *deletions*, *insertions*, and *substitutions*. These requirements are met by the *Translation Edit Rate* (TER) (Snover et al., 2006):

$$\mathrm{TER}(\mathbf{e}_i, \mathbf{e}_j) := \frac{\text{Del} + \text{Ins} + \text{Sub} + \text{Shift}}{|\mathbf{e}_j|} \tag{3}$$

The outcome of the iterated alignments is a word transition network which is also known as *word sausage* because of the linear sequence of correspondence sets that constitute the network. Since both the order and the elements of a correspondence set depend on the choice of the primary translation, each candidate translation is chosen in turn as the primary system. This results in a total of M word sausages that are combined into a single super network. The word sequence along the cost-minimizing path defines the composite translation.

To further optimize the word sausages, we replace each system prior weight ω_m with the l_p-norm over the normalized scalar product between the weight vector $\boldsymbol{\omega}$ and the row vector $\mathbf{b}_m$:

$$\omega'_m := \frac{(\boldsymbol{\omega}^\top \cdot \mathbf{b}_m)^\ell}{\sum_{\tilde{m}} (\boldsymbol{\omega}^\top \cdot \mathbf{b}_{\tilde{m}})^\ell}, \qquad \ell \in [0, +\infty) \tag{4}$$

As ℓ approaches $+\infty$, $\omega'_m = 1$ if and only if system m has the highest consensus among all input systems; otherwise, $\omega'_m = 0$. Thus, the word sausages are able to emulate the candidate selection rule described in Section 2.1. Setting $\ell = 0$ yields uniform system prior weights, and setting $\mathbf{B}$ to the unity matrix provides the original prior weights vector. Word sausages which take advantage of the refined system prior weights are denoted by *word sausages+*.

2.3 A Two-Pass Search Algorithm

The basic idea of the two-pass search algorithm is to compute a consensus translation by reordering words that are considered to be constituents of the final consensus translation.

Initially, the two-pass search is given a repository of candidate translations which serve as pseudo references together with a vector of system prior weights. In the first pass, the algorithm uses a greedy strategy to determine a *bag of words* which minimizes the *position-independent word error rate* (PER). These words are considered to be

[2] This property is not exclusive to this combination scheme but also holds for the methods discussed in Sections 2.2 and 2.3.

constituents of the final consensus translation. The greedy strategy implicitly ranks the constituents, i.e., words selected at the beginning of the first phase reduce the PER the most and are considered to be more important than constituents selected in the end. The first pass finishes when putting further constituents into the bag of words does not improve the PER.

The list of constituents is then passed to a second search algorithm, which starts with the empty string and then expands all active hypotheses by systematically inserting the next unused word from the list of constituents at different positions in the current hypothesis. For instance, a partial consensus hypothesis of length l expands into $l + 1$ new hypotheses of length $l + 1$. The resulting hypotheses are scored with respect to the TER measure based on the repository of weighted pseudo references. Low-scoring hypotheses are pruned to keep the space of active hypotheses small. The algorithm will finish if either no constituents are left or if expanding the set of active hypotheses does not further decrease the TER score. Optionally, the best consensus hypothesis found by the two-pass search is combined with all input translation systems via the MBR-like combination scheme described in Section 2.1. This refinement is called *two-pass+*.

2.4 Related Work

Research on multi-engine machine translation goes back to the early nineties. In (Robert and Nirenburg, 1994), a semi-automatic approach is described that combines outputs from three translation systems to build a consensus translation. (Nomoto, 2004) and (Paul et al., 2005) used translation scores, language and other models to select one of the original translations as consensus translation. (Bangalore et al., 2001) used a multiple string alignment algorithm in order to compute a single confusion network, on which a consensus hypothesis was computed through majority voting. Because the alignment procedure was based on the Levenshtein distance, it was unable to align translations with significantly different word orders. (Jayaraman and Lavie, 2005) tried to overcome this problem by using confidence scores and language models in order to rank a collection of synthetic combinations of words extracted from the original translation hypotheses. Experimental results were only reported for the METEOR metric (Banerjee and Lavie, 2005). In (Matusov et al., 2006), pairwise word alignments of the original translation hypotheses were estimated for an enhanced statistical alignment model in order to explicitly capture word re-ordering. Although the proposed method was not compared with other approaches to system combination, it resulted in substantial gains and provided new insights into system combination.

Table 1: *Corpus statistics for two Chinese-English text translation sets: ZHEN-05 is a random selection of test data used in NIST evaluations prior to 2006; ZHEN-06 comprises the NIST portion of the Chinese-English evaluation data used in the 2006 NIST machine translation evaluation.*

corpus		**Chinese**	**English**
ZHEN-05	sentences	2390	
	chars / words	110647	67737
ZHEN-06	sentences	1664	
	chars / words	64292	41845

3 Experimental Results

Experiments were conducted on two corpora for Chinese-English text translations, the first of which is compiled from a random selected subset of evaluation data used in the NIST MT evaluations up to the year 2005. The second data set consists of the NIST portion of the Chinese-English data used in the MT06 evaluation and comprises 1664 Chinese sentences collected from broadcast news articles (565 sentences), newswire texts (616 sentences), and news group texts (483 sentences). Both corpora provide 4 reference translations per source sentence. Table 1 summarizes some corpus statistics.

For all experiments, system performance was measured in terms of the IBM-BLEU score (Papineni et al., 2001). Compared to the NIST implementation of the BLEU score, IBM-BLEU follows the original definition of the brevity penalty (BP) factor: while in the NIST implementation the BP is always based on the length of the shortest reference translation, the BP in the IBM-BLEU score is based on the length of the reference translation which is closest to the candidate translation length. Typically, IBM-BLEU scores tend to be smaller than NIST-BLEU scores. In the following, BLEU always refers to the IBM-BLEU score.

Except for the results reported in Section 3.2, we used uniform system prior weights throughout all experiments. This turned out to be more stable when combining different sets of translation systems and helped to improve generalization.

Table 2: *BLEU scores and brevity penalty (BP) factors determined on the ZHEN-06 test set for primary systems together with consensus systems for the MBR-like candidate selection method obtained by combining each three adjacent systems with uniform system prior weights. Primary systems are sorted in descending order with respect to their BLEU score. The 95% confidence intervals are computed using the bootstrap re-sampling normal approximation method (Noreen, 1989).*

combination	**primary system**			**consensus**				**oracle**	
	BLEU	**CI 95%**	**BP**	**BLEU**	**Δ**	**BP**	**pair-CI 95%**	**BLEU**	**BP**
01, 02, 03	32.10	(±0.88)	0.93	32.97	(+0.87)	0.92	[+0.29, +1.46]	38.54	0.94
01, 15, 16*	32.10	(±0.88)	0.93	23.55	(-8.54)	0.92	[-9.29, -7.80]	33.55	0.95
02, 03, 04	31.71	(±0.90)	0.96	31.55	(-0.16)	0.92	[-0.65, +0.29]	37.23	0.95
03, 04, 05	29.59	(±0.88)	0.87	29.55	(-0.04)	0.88	[-0.53, +0.41]	35.55	0.92
03, 04, 06*	29.59	(±0.88)	0.87	29.83	(+0.24)	0.90	[-0.29, +0.71]	35.69	0.93
04, 05, 06	27.70	(±0.87)	0.94	28.52	(+0.82)	0.91	[+0.15, +1.49]	34.67	0.94
05, 06, 07	27.05	(±0.81)	0.88	28.21	(+1.16)	0.92	[+0.63, +1.66]	33.89	0.94
05, 06, 08*	27.05	(±0.81)	0.88	28.47	(+1.42)	0.91	[+0.95, +1.95]	34.18	0.93
06, 07, 08	27.02	(±0.76)	0.92	28.12	(+1.10)	0.94	[+0.59, +1.59]	33.87	0.95
07, 08, 09	26.75	(±0.79)	0.97	27.79	(+1.04)	0.94	[+0.52, +1.51]	33.54	0.95
08, 09, 10	26.41	(±0.81)	0.92	26.78	(+0.37)	0.94	[-0.07, +0.86]	32.47	0.96
09, 10, 11	25.05	(±0.84)	0.90	24.96	(-0.09)	0.94	[-0.59, +0.46]	30.92	0.97
10, 11, 12	23.48	(±0.68)	1.00	24.24	(+0.76)	0.94	[+0.27, +1.30]	30.08	0.96
11, 12, 13	23.26	(±0.74)	0.95	24.05	(+0.79)	0.92	[+0.40, +1.23]	29.56	0.93
12, 13, 14	22.38	(±0.78)	0.87	22.68	(+0.30)	0.89	[-0.28, +0.95]	28.58	0.91
13, 14, 15	22.13	(±0.72)	0.89	21.29	(-0.84)	0.90	[-1.33, -0.33]	26.61	0.92
14, 15, 16	17.42	(±0.66)	0.93	18.45	(+1.03)	0.92	[+0.45, +1.56]	23.30	0.95
15	17.20	(±0.64)	0.91	—	—	—	—	—	—
16	15.21	(±0.63)	0.96	—	—	—	—	—	—

3.1 Combining Multiple Research Systems

In a first experiment, we investigated the effect of combining translation outputs provided from different research labs. Each translation system corresponds to a primary system submitted to the NIST MT06 evaluation [3]. Table 2 shows the BLEU scores together with their corresponding BP factors for the primary systems of 16 research labs (site names were anonymized). Primary systems are sorted in descending order with respect to their BLEU score. Table 2 also shows the consensus translation results for the MBR-like candidate selection method. Except where marked with an asterisk, all consensus systems are built from the outputs of three adjacent systems. While only few combined systems show a degradation, the majority of all consensus translations achieve substantial gains between 0.2% and 1.4% absolute in terms of BLEU score on top of the best individual (primary) system. The column CI provides 95% confidence intervals for BLEU scores with respect to the primary system baseline using the bootstrap re-sampling normal approximation method (Noreen, 1989). The column "pair-CI" shows 95% confidence intervals relative to the primary system using the paired bootstrap re-sampling method (Koehn, 2004). The principle of the paired bootstrap method is to create a large number of corresponding virtual test sets by consistently selecting candidate translations with replacement from both the consensus and the primary system. The confidence interval is then estimated over the differences between the BLEU scores of corresponding virtual test sets. Improvements are considered to be significant if the left boundary of the confidence interval is larger than zero.

Oracle BLEU scores shown in Table 2 are computed by selecting the best translation among the three candidates. The oracle scores might indicate a larger potential of the MBR-like selection rule, and further gains could be expected if the candidate selection rule is combined with confidence measures.

Table 2 shows that it is important that all translation systems achieve nearly equal quality; combining high-performing systems with low-quality translations typically results in clear performance losses compared to the primary system, which is the case when combining, e.g., systems 01, 15, and 16.

[3] For more information see `http://www.nist.gov/speech/tests/mt/mt06eval_official_results.html`

Table 3: *BLEU scores and brevity penalty (BP) factors determined on the ZHEN-06 test set for the combination of multiple research systems using the MBR-like selection method with uniform and trained system prior weights. Prior weights are trained using 5-fold cross validation. The 95% confidence intervals realtive to uniform weights are computed using the paired bootstrap re-sampling method (Koehn, 2004).*

# systems	**combination**	**uniform**		**ω opt. on dev.**			**ω opt. on test**	
		BLEU	**BP**	**BLEU**	**BP**	**pair-CI 95%**	**BLEU**	**BP**
3	01 – 03	32.98	0.92	33.03	0.93	[-0.23, +0.34]	33.60	0.93
4	01 – 04	**33.44**	0.93	33.46	0.93	[-0.26, +0.29]	34.97	0.94
5	01 – 05	33.07	0.92	33.14	0.93	[-0.29, +0.43]	34.33	0.93
6	01 – 06	32.86	0.92	33.53	0.93	[+0.26, +1.08]	34.43	0.93
7	01 – 07	33.08	0.93	33.51	0.93	[+0.04, +0.82]	34.49	0.93
8	01 – 08	33.12	0.93	33.47	0.93	[-0.06, +0.75]	34.50	0.94
9	01 – 09	33.15	0.93	33.22	0.93	[-0.35, +0.51]	34.68	0.93
10	01 – 10	33.01	0.93	33.59	0.94	[+0.18, +0.96]	34.79	0.94
11	01 – 11	32.84	0.94	33.40	0.94	[+0.13, +0.98]	34.76	0.94
12	01 – 12	32.73	0.93	33.49	0.94	[+0.34, +1.18]	34.83	0.94
13	01 – 13	32.71	0.93	33.54	0.94	[+0.39, +1.26]	34.91	0.94
14	01 – 14	32.66	0.93	**33.69**	0.94	[+0.58, +1.47]	34.97	0.94
15	01 – 15	32.47	0.93	33.57	0.94	[+0.63, +1.57]	34.99	0.94
16	01 – 16	32.51	0.93	33.62	0.94	[+0.62, +1.59]	35.00	0.94

3.2 Non-Uniform System Prior Weights

As pointed out in Section 2.1, a useful property of the MBR-like system selection method is that system prior weights can easily be trained using the Minimum Error Rate Training (Och, 2003). In this section, we investigate the effect of using non-uniform system weights for the combination of multiple research systems. Since for each research system, only the first best translation candidate was provided, we used a five-fold cross validation scheme in order to train and evaluate the system prior weights. For this purpose, all research systems were consistently split into five random partitions of almost equal size. The partitioning procedure was document preserving, i.e., sentences belonging to the same document were guaranteed to be assigned to the same partition. Each of the five partitions played once the role of the evaluation set while the other four partitions were used as development data to train the system prior weights. Consensus systems were computed for each held out set using the system prior weights estimated on the respective development sets. The combination results determined on all held out sets were then concatenated and evaluated with respect to the ZHEN-06 reference translations. Table 3 shows the results for the combinations of up to 16 research systems using either uniform or trained system prior weights. System 01 achieved the highest BLEU score on all five constellations of development partitions and is therefore the primary system to which all results in Table 3 compare. In comparison to uniform weights, consensus translations using trained weights are more robust toward the integration of low performing systems into the combination scheme. The best combined system obtained with trained system prior weights (01-14) is, however, not significantly better than the best combined system using uniform weights (01-04), for which the 95% confidence interval yields $[-0.17, 0.66]$ according to the paired bootstrap re-sampling method.

Table 3 also shows the theoretically achievable BLEU scores when optimizing the system prior weights on the held out data. This provides an upper bound to what extent system combination might benefit if an ideal set of system prior weights were used.

3.3 Effect of Correlation on System Combination

The degree of correlation among input translation systems is a key factor which decides whether translation outputs can be combined such a way that the overall system performance improves. Correlation can be considered as a reciprocal measure of diversity: if the correlation is too large ($> 90\%$), there will be insufficient diversity among the input systems and the consensus system will at most be able to only marginally outperform the best indi-

Table 4: *BLEU scores obtained on ZHEN-05 with uniform prior weights and a 10-way system combination using the MBR-like candidate selection rule, word sausages, and the two-pass search algorithm together with their improved versions "sausages+" and "two-pass+", respectively for different sample sizes of the FBIS training corpus.*

sampling	**primary**			**mbr-like**		**sausages**		**sausages+**		**two-pass**		**two-pass+**	
[%]	**BLEU**	**CI 95%**	**BP**	**BLEU**	**BP**	**BLEU**	**BP**	**BLEU**	**BP**	**BLEU**	**BP**	**BLEU**	**BP**
5	27.82	(±0.65)	1.00	29.51	1.00	29.00	0.97	30.25	0.99	29.58	0.94	29.93	0.96
10	29.70	(±0.69)	1.00	31.42	1.00	30.74	0.98	31.99	0.99	31.30	0.95	31.75	0.97
20	31.37	(±0.69)	1.00	32.56	1.00	32.64	1.00	33.17	0.99	32.60	0.96	32.76	0.98
40	32.66	(±0.66)	1.00	33.52	1.00	33.23	0.99	33.98	1.00	33.65	0.97	33.88	0.99
80	33.67	(±0.66)	1.00	**34.17**	1.00	33.93	0.99	**34.38**	1.00	**34.20**	0.99	**34.35**	1.00
100	**33.90**	(±0.67)	1.00	34.03	1.00	**33.98**	1.00	34.02	1.00	33.90	1.00	34.08	1.00

vidual translation system. If the correlation is too low ($< 5\%$), there might be no consensus among the input systems and the quality of the consensus translations will hardly differ from a random selection of the candidates.

To study how correlation affects performance in system combination, we built a large number of systems trained on randomly sampled portions of the FBIS [4] training data collection. Sample sizes ranged between 5% and 100% with each larger data set doubling the size of the next smaller collection. For each sample size, we created 10 data sets, thus resulting in a total of 6×10 training corpora. On each data set, a new translation system was trained from scratch and used for decoding the ZHEN-05 test sentences. All 60 systems applied the MBR decision rule (Kumar and Byrne, 2004), which gave an additional 0.5% gain on average on top of using the *maximum a-posteriori* (MAP) decision rule. Systems trained on equally amounts of training data were incrementally combined. Figure 1 shows the evolution of the BLEU scores as a function of the number of systems as the sample size is increased from 5–100%. Table 4 shows the BLEU scores obtained with a 10-way system combination using the MBR-like candidate selection rule, word sausages, and the two-pass search algorithm together with their improved versions "sausages+" and "two-pass+", respectively. In order to measure the correlation between the individual translation systems, we computed the inter-system BLEU score matrix as shown exemplary

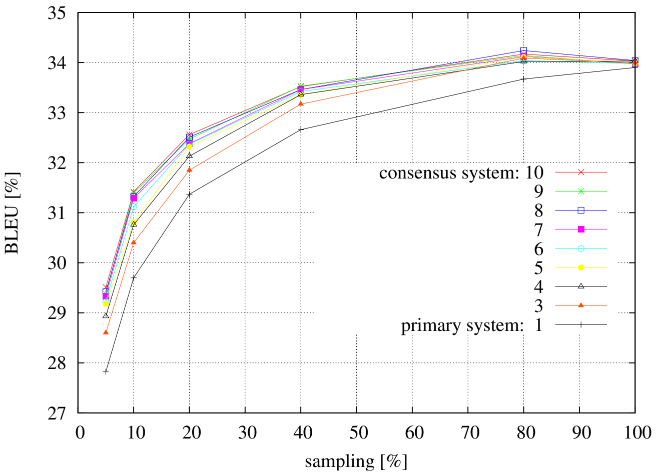


Figure 1: *Incremental system combination on ZHEN-05 using the MBR-like candidate selection rule and uniform prior weights. Systems were trained with different sample sizes of the FBIS data.*

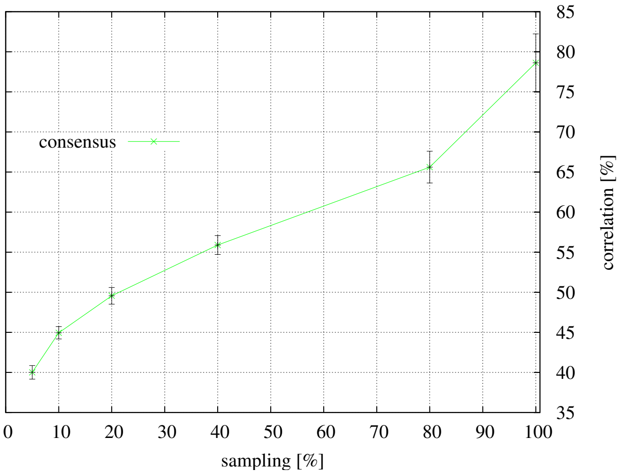


Figure 2: *Evolution of the correlation on ZHEN-05 averaged over 10 systems in the course of the sample size.*

[4] LDC catalog number: LDC2003E14

Table 5: *Minimum, maximum, and average inter-system BLEU score correlations for (i) the primary systems of the 2006 NIST machine translation evaluation on the ZHEN-06 test data, (ii) different training corpus sizes (FBIS), and (iii) a greedy strategy which chooses 15 systems out of a pool of 200 translation systems.*

	ZHEN-6	**ZHEN-5**						**ZHEN-5**	**ZHEN-6**
	16 primary	FBIS sampling,	10 systems					15 systems	15 systems
	systems	5%	10%	20%	40%	80%	100%	greedy selection	ZHEN-5 selection
min	0.08	0.38	0.44	0.47	0.53	0.60	0.72	0.55	0.50
mean	0.18	0.40	0.45	0.50	0.56	0.66	0.79	0.65	0.61
median	0.19	0.40	0.45	0.49	0.56	0.64	0.78	0.63	0.58
max	0.28	0.42	0.47	0.53	0.58	0.70	0.88	0.85	0.83

in Table 6 for the 16 MT06 primary submissions. Figure 2 shows the evolution of the correlation averaged over 10 systems as the sample size is increased from 5–100%. Note that all systems were optimized using a non-deterministic implementation of the *Minimum Error Rate Training* described in (Och, 2003). Hence, using all of the FBIS corpus data does not necessarily result in fully correlated systems, since the training procedure may pick a different solution for same training data in order to increase diversity. Both Table 4 and Figure 1 clearly indicate that increasing the correlation (and thus reducing the diversity) substantially reduces the potential of a consensus system to outperform the primary translation system. Ideally, the correlation should not be larger than 30%.

Especially for low inter-system correlations and reduced translation quality, both the enhanced versions of the word sausage combination method and the two-pass search outperform the MBR-like candidate selection scheme. This advantage, however, diminishes as soon as the correlation increases and translations produced by the individual systems become more similar.

3.4 Toward Automatic System Generation and Selection

Sampling the training data is an effective means to investigate the effect of system correlation on consensus performance. However, this is done at the expense of the overall system quality. What we need instead is a method to reduce correlation without sacrificing system performance.

A simple, though computationally very expensive way to build an ensemble of low-correlated statistical machine translation systems from a single translation engine is to train a large pool of systems, in which each of the systems is trained with a slightly different set of parameters. Changing only few parameters at a time typically results in only small changes in system performance but may have a strong impact on system correlation. In our experiments we observed that changing parameters which affect the training procedure at a very early stage, are most effective and introduce larger diversity. For instance, changing the training procedure for word alignment models turned out to be most beneficial; for details see (Och and Ney, 2003). Other parameters that were changed include the maximum jump width in word re-ordering, the choice of feature function weights for the log-linear translation models, and the set of language models used in decoding.

Once a large pool of translation systems has been generated, we need a method to select a small ensemble of diverse translation outputs that are beneficial for computing consensus translations. Here, we used a greedy strategy to rank the systems with respect to their ability to improve system

Table 6: *Inter-system BLEU score matrix for primary systems of the NIST 2006 TIDES machine translation evaluation on the ZHEN-06 test data.*

Id	**01**	**02**	**03**	**04**	**05**	···	**14**	**15**	**16**
01	1.00	0.27	0.26	0.23	0.26	···	0.15	0.15	0.12
02	0.27	1.00	0.27	0.22	0.25	···	0.15	0.15	0.12
03	0.26	0.27	1.00	0.21	0.28	···	0.15	0.15	0.10
04	0.23	0.22	0.21	1.00	0.19	···	0.14	0.12	0.12
05	0.26	0.25	0.28	0.19	1.00	···	0.16	0.17	0.11
06	0.27	0.24	0.25	0.21	0.26	···	0.16	0.18	0.13
⋮						⋱			⋮
14	0.15	0.15	0.15	0.14	0.16	···	1.00	0.12	0.08
15	0.15	0.15	0.15	0.12	0.17	···	0.12	1.00	0.09
16	0.12	0.12	0.10	0.12	0.11	···	0.08	0.09	1.00

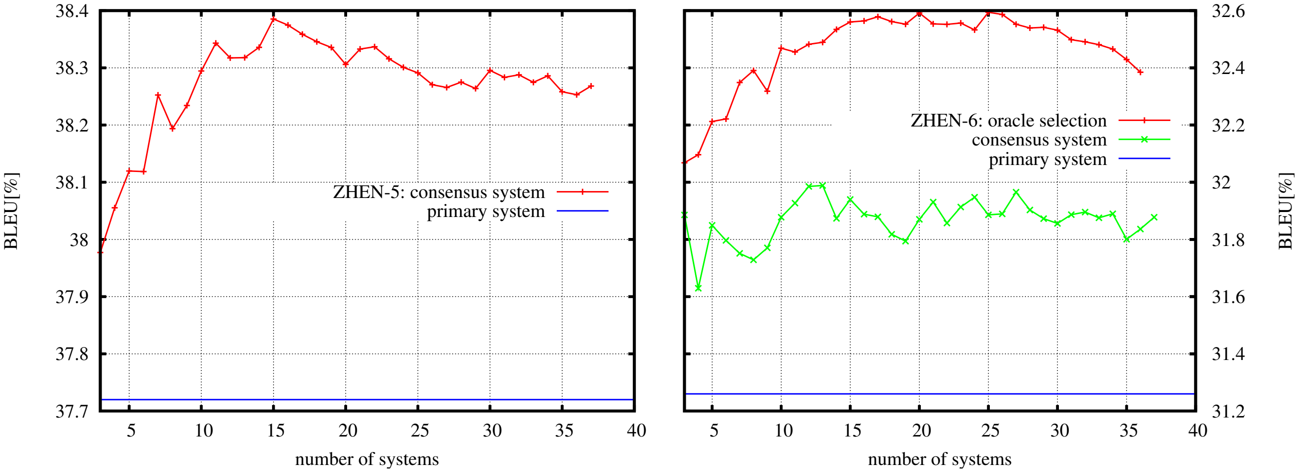


Figure 3: *BLEU score of the consensus translation as a function of the number of systems on the ZHEN-05 sentences (left) and ZHEN-06 sentences (right). The middle curve (right) shows the variation of the BLEU score on the ZHEN-06 data when the greedy selection of the ZHEN-05 is used.*

combination. Initially, the greedy strategy selected the best individual system and then continued by adding those systems to the ensemble, which gave the highest gain in terms of BLEU score according to the MBR-like system combination method. Note that the greedy strategy is not guaranteed to increase the BLEU score of the combined system when a new system is added to the ensemble of translation systems.

In a first experiment, we trained approximately 200 systems using different parameter settings in training. Each system was then used to decode both the ZHEN-05 and the ZHEN-06 test sentences using the MBR decision rule. The upper curve in Figure 3 (left) shows the evolution of the BLEU score on the ZHEN-05 sentences in the course of the number of selected systems. The upper curve in Figure 3 (right) shows the BLEU score of the consensus translation as a function of the number of systems when the selection is done on the ZHEN-06 set. This serves as an oracle. The middle curve (right) shows the function of the BLEU score when the system selection made on the ZHEN-05 set is used in order to combine the translation outputs for the ZHEN-06 data. Although system combination gave moderate improvements on top of the primary system, the greedy strategy still needs further refinements in order to improve generalization. While the correlation statistics shown in Table 5 indicate that changing the training parameters helps to substantially decrease system correlation, there is still need for additional methods in order to reduce the level of inter-system BLEU scores such that they fall within the range of $[0.2, 0.3]$.

4 Conclusions

In this paper, we presented an empirical study on how different selections of translation outputs affect translation quality in system combination. Composite translations were computed using (i) a candidate selection method based on inter-system BLEU score matrices, (ii) an enhanced version of word sausage networks, and (iii) a novel two-pass search algorithm which determines and re-orders bags of words that build the constituents of the final consensus hypothesis. All methods gave statistically significant improvements.

We showed that both a high diversity among the original translation systems and a similar translation quality among the translation systems are essential in order to gain substantial improvements on top of the best individual translation systems.

Experiments were conducted on the NIST portion of the Chinese English text translation corpus used for the 2006 NIST machine translation evaluation. Combined systems were built from primary systems of up to 16 different research labs as well as systems derived from one of the best-ranked translation engines.

We trained a large pool of translation systems from a single translation engine and presented first experimental results for a greedy search to select an ensemble of translation systems for system combination.

References

S. Banerjee and A. Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization, 43th Annual Meeting of the Association of Computational Linguistics (ACL-2005)*, Ann Arbor, MI, USA, June.

S. Bangalore, G. Bodel, and G. Riccardi. 2001. Computing Consensus Translation from Multiple Machine Translation Systems. In *2001 Automatic Speech Recognition and Understanding (ASRU) Workshop*, Madonna di Campiglio, Trento, Italy, December.

T. Brants, A. Popat, P. Xu, F. Och, and J. Dean. 2007. Large Language Models in Machine Tranlation. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing*, Prague, Czech Republic. Association for Computational Linguistics.

J. G. Fiscus. 1997. A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER). In *Proceedings 1997 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–352, Santa Barbara, CA, USA, December.

S. Jayaraman and A. Lavie. 2005. Multi-Engine Machine Translation Guided by Explicit Word Matching. In *10th Conference of the European Association for Machine Translation (EAMT)*, pages 143–152, Budapest, Hungary.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Edmonton, Canada. Association for Computational Linguistics.

P. Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, August. Association for Computational Linguistics.

S. Kumar and W. Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proc. HLT-NAACL*, pages 196–176, Boston, MA, USA, May.

E. Matusov, N. Ueffing, and H. Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–40, Trento, Italy, April.

T. Nomoto. 2004. Multi-Engine Machine Translation with Voted Language Model. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 494–501, Barcelona, Spain, July.

E. W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons, Canada.

F. J. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

F. J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, USA.

M. Paul, T. Doi, Y. Hwang, K. Imamura, H. Okuma, and E. Sumita. 2005. Nobody is Perfect: ATR's Hybrid Approach to Spoken Language Translation. In *International Workshop on Spoken Language Translation*, pages 55–62, Pittsburgh, PA, USA, October.

F. Robert and S. Nirenburg. 1994. Three Heads are Better than One. In *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing*, Stuttgart, Germany, October.

K. C. Sim, W. Byrne, M. Gales, H. Sahbi, and P.C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Honolulu, HI, USA, April.

M. Snover, B. J. Dorr, R. Schwartz, J. Makhoul, L. Micciulla, and R. Weischedel. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Learning to Find English to Chinese Transliterations on the Web

Jian-Cheng Wu
Department of Computer Science
National Tsing Hua University
101, Kuangfu Road, Hsinchu, Taiwan
d928322@oz.nthu.edu.tw

Jason S. Chang
Department of Computer Science
National Tsing Hua University
101, Kuangfu Road, Hsinchu, Taiwan
jschang@cs.nthu.edu.tw

Abstract

We present a method for learning to find English to Chinese transliterations on the Web. In our approach, proper nouns are expanded into new queries aimed at maximizing the probability of retrieving transliterations from existing search engines. The method involves learning the sublexical relationships between names and their transliterations. At run-time, a given name is automatically extended into queries with relevant morphemes, and transliterations in the returned search snippets are extracted and ranked. We present a new system, *TermMine*, that applies the method to find transliterations of a given name. Evaluation on a list of 500 proper names shows that the method achieves high precision and recall, and outperforms commercial machine translation systems.

1 Introduction

Increasingly, short passages or web pages are being translated by desktop machine translation software or are submitted to machine translation services on the Web every day. These texts usually contain some proportion of proper names (e.g., place and people names in “The cities of Mesopotamia prospered under Parthian and Sassanian rule.”), which may not be handled properly by a machine translation system. Online machine translation services such as *Google Translate*[1] or *Yahoo! Babelfish*[2] typically use a bilingual dictionary that is either manually compiled or learned from a parallel corpus. However, such dictionaries often have insufficient coverage of proper names and technical terms, leading to poor translation performance due to out of vocabulary (OOV) problem.

Handling name transliteration is also important for cross language information retrieval (CLIR) and terminology translation (Quah 2006). There are also services on the Web specifically targeting transliteration aimed at improving CLIR, including *CHINET* (Kwok et al. 2005) and *LiveTrans* (Lu, Chien, and Lee 2004).

The OOV problems of machine translation (MT) or CLIR can be handled more effectively by learning to find transliteration on the Web. Consider the sentence in Example (1), containing three proper names. *Google Translate* produces the sentence in Example (2) and leaves “Parthian” and “Sassanian” not translated. A good response might be a translation like Example (3) with appropriate transliterations (underlined).

(1) *The cities of Mesopotamia prospered under Parthian and Sassanian rule.*

(2) 城市繁榮下 parthian 達米亞、sassanian 統治。

(3) <u>美索不達米亞</u>[3]城市在<u>巴底亞</u>[4]和<u>薩珊</u>[5]統治下繁榮起來。

These transliterations can be more effectively retrieved from mixed-code Web pages by extending each of the proper names into a query. Intuitively, by requiring one of likely transliteration morphemes (e.g., “巴”(Ba) or “帕”(Pa) for names beginning with the prefix “par-”), we can bias the search engine towards retrieving the correct trans-

[1] Google Translate: translate.google.com/translate_t

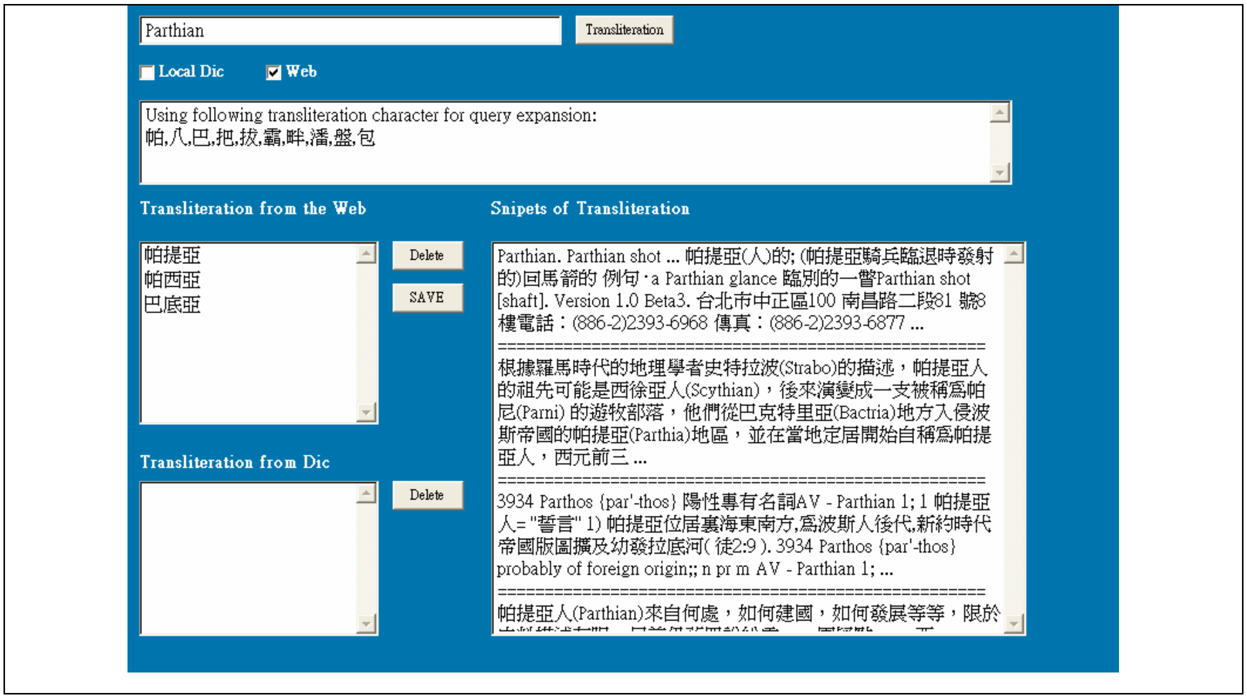
[2] Yahoo! Babelfish: babelfish.yahoo.com

[3] 美索不達米亞(Meisuobudamiya) is the transliteration of “Mesopotamia.”

[4] 巴底亞(Badiya) is the transliteration of “Parthian.”

[5] 薩珊(Sashan) is the transliteration of “Sassanian.”

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 996–1004, Prague, June 2007.



Parthian

☐ Local Dic ☑ Web

Using following transliteration character for query expansion:
帕,八,巴,把,拔,霸,畔,潘,盤,包

Transliteration from the Web

帕提亞
帕西亞
巴底亞

Transliteration from Dic

Snipets of Transliteration

Parthian. Parthian shot ... 帕提亞(人)的; (帕提亞騎兵臨退時發射的)回馬箭的 例句·a Parthian glance 臨別的一瞥Parthian shot [shaft]. Version 1.0 Beta3. 台北市中正區100 南昌路二段81 號8樓電話：(886-2)2393-6968 傳真：(886-2)2393-6877 ...

===

根據羅馬時代的地理學者史特拉波(Strabo)的描述，帕提亞人的祖先可能是西徐亞人(Scythian)，後來演變成一支被稱爲帕尼(Parni) 的遊牧部落，他們從巴克特里亞(Bactria)地方入侵波斯帝國的帕提亞(Parthia)地區，並在當地定居開始自稱爲帕提亞人，西元前三 ...

===

3934 Parthos {par'-thos} 陽性專有名詞AV - Parthian 1; 1 帕提亞人= "誓言" 1) 帕提亞位居裏海東南方,爲波斯人後代,新約時代帝國版圖擴及幼發拉底河(徒2:9). 3934 Parthos {par'-thos} probably of foreign origin;; n pr m AV - Parthian 1; ...

===

帕提亞人(Parthian)來自何處，如何建國，如何發展等等，限於

Figure 1. An example of *TermMine* search for transliterations of the name “Parthian”

literations (e.g., “巴底亞”(Badiya) and “帕提亞”(Patiya)) in snippets of many top-ranked documents.

This approach to terminology translation by searching is a strategy increasingly adopted by human translators. Quah (2006) described a modern day translator would search for the translation of a difficult technical term such as “異方性導電樹脂フィルム” by expanding the query with the word “*film*” (back transliteration of the component “フィルム” of the term in question). This kind of query expansion (QE) indeed increases the chance of finding the correct translation “*anisotropic conductive film*” in top-ranked snippets. However, the manual process of expanding query, sending search request, and extracting transliteration is tedious and time consuming. Furthermore, unless the query expansion is done properly, snippets containing answers might not be ranked high enough for this strategy to be the most effective.

We present a new system, *TermMine*, that automatically learns to extend a given name into a query expected to retrieve and extract transliterations of the proper name. An example of machine transliteration of “Parthian” is shown in Figure 1. *TermMine* has determined the best 10 query expansions (e.g., “Parthian 巴,” “Parthian 帕”). *TermMine* learns these effective expansions automatically during training by analyzing a collection of place names and their transliterations, and deriving cross-language relationships of prefix and postfix morphemes. For instance, *TermMine* learns that a name that begins with the prefix “par-” is likely to have a transliteration beginning with “巴” or “帕”). We describe the learning process in Section 3.

This prototype demonstrates a novel method for learning to find transliterations of proper nouns on the Web based on query expansion aimed at maximizing the probability of retrieving transliterations from existing search engines. Since the method involves learning the morphological relationships between names and their transliterations, we refer to this IR-based approach as *morphological query expansion approach to machine transliteration*. This novel approach is general in scope and can also be applied to *back transliteration* and to *translation* with slight modifications, even though we focus on transliteration in this paper.

The remainder of the paper is organized as follows. First, we give a formal statement for the problem (Section 2). Then, we present a solution to the problem by proposing new transliteration probability functions, describing the procedure for estimating parameters for these functions (Section 3) and the run-time procedure for searching and extracting transliteration via a search engine (Section 4). As part of our evaluation, we carry out two sets of experiments, with or without query expansion, and compare the results. We also evaluate the results against two commercial machine translation online services (Section 5).

2 Problem Statement

Using online machine translation services for name transliteration does not work very well. Searching in the vicinity of the name in mixed-code Web pages is a good strategy. However, query expansion is needed for this strategy to be effective. Therefore, to find transliterations of a name, a promising approach is to automatically expand the given name into a query with the additional requirement of some morpheme expected to be part of relevant transliterations that might appear on the Web.

Table 1. Sample name-transliteration pairs from the training collection.

Name	Transliteration	Name	Transliteration
Aabenraa	阿本洛	Aarberg	阿爾柏
Aabybro	阿比布洛	Aarburg	亞爾堡
Aachen	亞琛	Aardenburg	亞丁堡
Aalesund	奧勒孫	Aargau	亞高
Aaley	阿利	Aars	阿爾斯
Aalten	阿爾廷	Aba	阿巴
Aarau	亞牢	Abacaxis	阿巴卡克斯

Now, we formally state the problem we are dealing with:

> While a *proper name N* is given. Our goal is to search and extract the *transliteration T* of *N* from Web pages via a general-purpose *search engine SE*. For that, we expand *N* into a set of queries q_1, q_2, …, q_m, such that the top *n* document snippets returned by *SE* for the queries are likely to contain some transliterations *T* of the given name *N*.

In the next section, we propose using a probability function to model the relationships between *names* and *transliterations* and describe how the parameters in this function can be estimated.

3 Learning Relationships for QE

We attempt to derive cross-language morphological relationships between names and transliterations and use them to expand a name into an effective query for searching and extracting transliterations. For the purpose of expanding the given name, *N*, into effective queries to search and extract transliterations *T*, we define a probabilistic function for mapping prefix syllable from the source to the target languages. The prefix transliteration function $P(T_P \mid N_P)$ is the probability of *T* has a prefix T_P under the condition that the name *N* has a prefix N_P.

$$P\,(T_P \mid N_P) = \text{Count}\,(T_P,N_P)\,/\,\text{Count}\,(N_P) \qquad (1)$$

where Count (T_P,N_P) is the number of T_P and N_P co-occurring in the pairs of training set (see Table 1), and Count(N_P) is the number of N_P occurring in training set.

Similarly, we define the function $P\,(T_S \mid N_S)$ for postfixes T_S and N_S:

$$P\,(T_S \mid N_S) = \text{Count}\,(T_S,N_S)\,/\,\text{Count}\,(N_S) \qquad (2)$$

The prefixes and postfixes are intended as a syllable in the two languages involved, so the two prefixes correspond to each other (See Table 2&3). Due to the differences in the sound inventory, the Roman prefix corresponding to a syllabic prefix in Chinese may vary, ranging from a consonant, a vowel, or a consonant followed by a vowel (but not a vowel followed by a consonant). So, it is likely such a Roman prefix has from one to four letters. On the contrary, the prefix syllable for a name written in Chinese is readily identifiable.

Table 2. Sample cross-language morphological relationships between prefixes.

Name Prefix (N_P)	**Transliteration Prefix (T_P)**	**N_P Count**	**T_P Count**	**Co-occ. Count**
a-	阿(A)	1,456	854	854
a-	亞(Ya)	1,456	267	264
ab-	阿(A)	77	854	45
ab-	亞(Ya)	77	267	32
b-	布(Bu)	2,319	574	566
b-	巴(Ba)	2,319	539	521
ba-	巴(Ba)	650	574	452
bu-	布(Bu)	299	539	182

Table 3. Sample cross-language morphological relationships between postfixes.

Name Postfix (N_s)	Transliteration Postfix (T_s)	N_s Count	T_s Count	Co-occ. Count
-a	拉(La)	4,774	1,044	941
-a	亞(Ya)	4,774	606	568
-la	拉(La)	461	1,044	422
-ra	拉(La)	534	1,044	516
-ia	亞(Ya)	456	606	391
-nia	亞(Ya)	81	606	77
-burg	堡(Bao)	183	230	175

We also observe that a preferred prefix (e.g., “艾”(Ai)) is often used for a Roman prefix (e.g., “a-” or “ir-”), while occasionally other homophonic characters are used (e.g., “埃”(Ai)). The skew distribution creates problems for reliable estimation of transliteration functions. To cope with this data sparseness problem, we use homophone classes and a function CL that maps homophonic characters to the same class number. For instance, “艾” and “埃” are homophonic, and both are assigned the same class identifier(see Table 4 for more samples).

Therefore, we have

CL (“艾”) = CL (“埃”) = 275.

Table 4. Some examples of classes of homophonic characters. The class ID of each class is assigned arbitrarily.

Class ID	Transl. char	Pronunciation	Class ID	Transl. char	Pronunciation
1	八	Ba	2	波	Bo
1	巴	Ba	275	艾	Ai
1	拔	Ba	275	埃	Ai
1	把	Ba	275	愛	Ai
1	罷	Ba	276	敖	Ao
1	霸	Ba	276	奧	Ao
2	白	Bo	276	澳	Ao
2	伯	Bo	…	…	…

With homophonic classes of transliteration morphemes, we define class-based transliteration probability as follows

$$P_{CL}(C\,|\,N_P) = \text{Count}(T_P,N_P)\,/\,\text{Count}(N_P) \qquad (3)$$

where $CL(T_P) = C$

$$P_{CL}(C\,|\,N_S) = \text{Count}(T_S,N_S)\,/\,\text{Count}(N_S) \qquad (4)$$

where $CL(T_S) = C$

and then we rewrite P ($T_P\,|\,N_P$) and P ($T_S\,|\,N_S$) as

$$P\,(T_P\,|\,N_P) = P_{CL}(CL(T_P\,)\,|\,N_P) \qquad (5)$$

$$P\,(T_S\,|\,N_S) = P_{CL}(CL(T_S\,)\,|\,N_S) \qquad (6)$$

With class-based transliteration probabilities, we are able to cope with difficulty in estimating parameters for rare events which are under represented in the training set. Table 5 shows that “埃” belongs to a homophonic class co-occurring with “a-” for 46 times, even when only one instance of (“埃”, “a-”).

After cross-language relationships for prefixes and postfixes are automatically trained, the prefix relationships are stored as prioritized query expansion rules. In addition to that, we also need a transliteration probability function to rank candidate transliterations at run-time (Section 4). To cope with data sparseness, we consider names (or transliterations) with the same prefix (or postfix) as a class. With that in mind, we use both prefix and postfix to formulate an interpolation-based estimator for name transliteration probability:

$$P(T\,|\,N)=\max_{N_P,\,N_S}\ \lambda_1 P(T_P\,|\,N_P)+\lambda_2 P(T_S\,|\,N_S) \qquad (7)$$

where $\lambda_1 + \lambda_2 = 1$ and N_P, N_S, T_P, and T_S are the prefix and postfix of the given name N and transliteration T.

For instance, the probability of “美索不達米亞”(Meisuobudamiya) as a transliteration of “*Mesopotamia*” is estimated as follows

P (美索不達米亞 | “*Mesopotamia*”)
= λ_1P (“美” | “me-”)+ λ_2 P (“亞” | “-a”)

(1) For each entry in the bilingual name list, pair up prefixes and postfixes in names and transliterations.
(2) Calculate counts of these affixes and their co-occurrences.
(3) Estimate the prefix and postfix transliteration functions
(4) Estimate class-based prefix and postfix transliteration functions

Figure 2. Outline of the process used to train the *TermMine* system.

The system follows the procedure shown in Figure 2 to estimate these probabilities. In Step (1), the system generates all possible prefix pairs for each name-transliteration pair. For instance, consider the pair, (“Aabenraa,” “阿本洛”), the system will generate eight pairs:

(a-, 阿-), (aa-, 阿-), (aab-, 阿-), (aabe-, 阿-),
(-a, -洛), (-aa, -洛), (-raa, -洛), and (-nraa, -洛).

Finally, the transliteration probabilities are estimated based on the counts of prefixes, postfixes, and their co-occurrences. The derived probabilities embody a number of relationships:

(a) Phoneme to syllable relationships (e.g., “b” vs. “布” as in “Brooklyn” and “布魯克林”(Bulukelin)),

(b) Syllable to syllable relationships (e.g., “bu” vs. “布”),

(c) Phonics rules (e.g., “br-“ vs. “布” and “克” vs. “cl-”). The high probability of P(“克” | “cl-”) amounts to the phonics rule that stipulates “c” be pronounced with a “k” sound in the context of “l.”

4 Transliteration Search and Extraction

At run-time, the system follows the procedure in Figure 3 to process the given name. In Step (1), the system looks up in the prefix relationship table to find the *n* best relationships (*n* = *MaxExpQueries*) for query expansion with preference for relationships with higher probabilistic value. For instance, to search for transliterations of “Acton,” the system looks at all possible prefixes and postfixes of “Acton,” including *a-*, *ac-*, *act-*, *acto-*, *-n*, *-on*, *-ton*, and *-cton*, and determines the best query expansions: “Acton 阿,” “Acton 亞,” “Acton 艾,” “Acton 頓,” “Acton 騰,” etc. These effective expansions are automatically derived during the training stage described in Section 3 by analyzing a large collection of name-transliteration pairs.

In Step (2), the system sends off each of these queries to a search engine to retrieve up to *MaxDocRetrieved* document snippets. In Step (3), the system discards snippets that have too little proportion of target-language text. See Example (4) for a snippet that has high portion of English text and therefore is less likely to contain a transliteration. In Step (4), the system considers the substrings in the remaining snippets.

(1) Look up the table for top *MaxExpQueries* prefix and posfix relationships relevant to the given name and use the target morphemes in the relationship to form expanded queries
(2) Search for Web pages with the queries and filter out snippets containing at less than *MinTargetRate* portion of target language text
(3) Evaluate candidates based on class-based transliteration probability (Equation 5)
(4) Output top one candidate for evaluation

Figure 3. Outline of the steps used to search, extract, and rank transliterations.

Table 5. Sample data for class-based morphological transliteration probability of prefixes, where # of N_P denotes the number of the name prefix N_P; # of C, N_P denotes the number of all T_P belonging to the class C co-occurring with the N_P; # T_P, N_P denotes the number of transliteration prefix T_P co-occurs with the N_P; P(C|N_P) denotes the probability of all T_P belonging to C co-occurring with the N_P; P(T_P|N_P) denotes the probability of the T_p co-occurs with the N_P.

N_P	Class ID	T_P	# of N_P	# of C,N_P	# of T_P,N_P	P(C|N_P)	P(T_P|N_P)
a-	275	艾	1456	46	28	0.032	0.019
a-	275	愛	1456	46	17	0.032	0.012
a-	275	埃	1456	46	1	0.032	0.000
a-	276	奧	1456	103	100	0.071	0.069
a-	276	澳	1456	103	2	0.071	0.001
a-	276	敖	1456	103	1	0.071	0.000
ba-	2	波	652	5	3	0.008	0.005
ba-	2	百	652	5	1	0.008	0.002
ba-	2	柏	652	5	1	0.008	0.002

Table 6. Sample data for class-based morphological transliteration probability of postfixes. Notations are similar to those for Table 5.

N_s	Class ID	T_s	# of N_s	# of C,N_s	# of T_s,N_s	P(C|N_s)	P(T_s|N_s)
-li	103	利	142	140	85	0.986	0.599
-li	103	里	142	140	52	0.986	0.366
-li	103	力	142	140	2	0.986	0.014
-li	103	立	142	140	1	0.986	0.007
-li	103	李	142	140	0	0.986	0.000
-raa	112	洛	4	1	1	0.250	0.250
-raa	112	珞	4	1	0	0.250	0.000
-raa	112	絡	4	1	0	0.250	0.000
-raa	112	落	4	1	0	0.250	0.000

For instance, Examples (5-7) shows remaining snippets that have high proportion of Chinese text. The strings “阿克頓”(Akedun) is a transliteration found in snippet shown in Example (5), a candidate beginning with the prefix “阿” and ending with the postfix “頓” and is within the distance of 1 of the instance “Acton,” separated by a punctuation token. The string “埃克頓” (Aikedun) found in Example (6) is also a legitimate transliteration beginning with a different prefix “埃, ” while “艾科騰”(Aiketeng) in Example (7) is a transliteration beginning with yet another prefix “艾.” Transliteration “埃克頓” appears at a distance of 3 from “Acton,” while two instances of “艾科騰” appear at the distances of 1 and 20 from the nearest instances of “Acton.”

(4) Acton moive feel pics!! - 攝影
目前位置: 文藝線 > 遊藝支線 > 攝影 > Acton moive feel pics!! Hop Hero - Acton moive feel pics!!
http://www.hkmassive.com/forum/viewthread.php?tid=2368&fpage=1 Watch the slide show! ...

(5) New Home Alert - Sing Tao New Homes
Please select, Acton 阿克頓, Ajax 亞積士, Alliston 阿里斯頓, Ancaster 安卡斯特, Arthur 阿瑟, Aurora 奧羅拉, Ayr 艾爾, Barrie 巴里, Beamsville, Belleville ...

(6) STS-51-F – Wikipedia
前排左起：英格蘭、海因茲、福勒頓、布里奇斯 ... 卡爾·海因茲 (Karl Henize，曾執行 STS-51-F任務)，任務專家; 羅倫·埃克頓 (Loren Acton，曾執行 STS-51-F 任務)，有效載荷專家; 約翰-大衛·巴托 (John-David F. ...

(7) 澳洲艾科騰-00-Acton-Australia.htm
Acton Systems is a world leading manufacturer supplying stuctured cabling systems suited to the Australian and New Zealand marketplace. 澳洲艾科騰乃專業之整合式配線系統製造商, 產品銷售於澳洲及紐西蘭。 Custom made leads are now available ...

The occurrence counts and average distance from instances of the given name are tallied for each of these candidates. Candidates with a low occurrence count and long average distance are excluded from further consideration. Finally, all candidates are evaluated and ranked using Equation (7) given in Section 3.

5 Evaluation

In the experiment carried out to assess the feasibility to the proposed method, a data set of 23,615 names and transliterations was used. This set of place name data is available from NICT, Taiwan for training and testing. There are 967 distinct Chinese characters presented in the data, and more details of training data are available in Table 7. The English part consists of Romanized versions of names originated from many languages, including Western and Asian languages. Most of the time, the names come with a Chinese counterpart based solely on transliteration. But occasionally, the Chinese counterpart is part translation and part transliteration. For instance, the city of “Southampton” has a Chinese counterpart consisting of “南” (translation of “south”) and “漢普頓” (transliteration of “ampton”).

Table 7. Training data and statistics

Type of Data Used in Experiment	Number
Name-transliteration pairs	23,615
Training data	23,115
Test data	500
Distinct transliteration morphemes	967
Distinct transliteration morphemes (80% coverage)	100
Names with part translation and part transliteration (estimated)	300
Cross-language prefix relationships	21,016
Cross-language postfix relationships	26,564

We used the set of parameters shown in Table 8 to train and run System *TermMine*. A set of 500 randomly selected were set aside for testing. We paired up the prefixes and postfixes in the remaining 23,116 pairs, by taking one to four leading or trailing letters of each Romanized place names and the first and last Chinese transliteration character to estimate P ($T_P | N_P$) and P ($T_S | N_S$).

Table 8. Parameters for training and testing

Parameter	Value	Description
MaxPrefixLetters	4	Max number of letters in a prefix
MaxPostfixLetters	4	Max number of letters in a postfix
MaxExpQueries	10	Max number of expanded queries
MaxDocRetrieved	1000	Max number of document retrieved

MinTargetRate	0.5	Min rate of target text in a snippet
MinOccCount	1	Min number of co-occurrence of query and transliteration candidate in snippets
MaxAvgDistance	4	Max distance between *N* and *T*
WeightPrefixProb	0.5	Weight of Prefix probability (λ_1)
WeightPostfixProb	0.5	Weight of Postfix probability (λ_2)

We carried out two kinds of evaluation on System *TermMine*, with and without query expansion. With QE option off, the name itself was sent off as a query to the search engine, while with QE option turned on, up to 10 expanded queries were sent for each name. We also evaluated the system against *Google Translate* and *Yahoo! Babelfish*. We discarded the results when the names are returned untranslated. After that, we checked the correctness of all remaining results by hand. Table 9 shows a sample of the results produced by the three systems.

In Table 10, we show performance differences of system *TermMine* in query expansion option. Without QE, the system returns transliterations (applicability) less than 50% of the time. Nevertheless, there are enough snippets for extracting and ranking of transliterations. The precision rate of the top-ranking transliterations is 88%. With QE turned on, the applicability rate increases significantly to 60%. The precision rate also improved slightly to 0.89.

The performance evaluation of three systems is shown in Table 11. For the test set of 500 place names, *Google Translate* returned 146 transliterations and *Yahoo! Babelfish* returned only 44, while *TermMine* returned 300. Of the returned transliterations, *Google Translate* and *Yahoo! Babelfish* achieved a precision rate around 50%, while *TermMine* achieved a precision rate almost as high as 90%. The results show that System *TermMine* outperforms both commercial MT systems by a wide margin, in the area of machine transliteration of proper names.

Table 9. Sample output by three systems evaluated. The stared transliterations are incorrect.

Name	***TermMine***	***Google Translate***	***Yahoo! Babelfish***
Arlington	雅靈頓	阿靈頓	阿靈頓
Toledo	托雷多	托萊多	-
Palmerston	帕默斯頓	帕麥斯頓	-
Cootamundra	庫塔曼德拉	庫塔曼德拉	-
Bangui	班基	班吉	-
Australasia	澳大拉西亞	*大洋洲	澳大利西亞
Wilson	威爾森	威爾遜	威爾遜
Mao	*馬寅卯	毛	毛
Inverness	因弗內斯	*禮士	因弗內斯
Cyprus	賽普勒斯	賽普勒斯	塞浦路斯
Rostock	羅斯托克	羅斯托克	羅斯托克
Bethel	貝瑟爾	貝瑟爾	*聖地
Arcade	阿凱德	*商場	*拱廊
Lomonosov	羅蒙諾索夫	羅蒙諾索夫	-
Oskaloosa	奧斯卡盧薩	奧斯卡羅薩	-

Table 10. Performance evaluation of TermMine

Method Evaluation	TermMine QE-	TermMine QE+
# of cases performed	238	300
Applicability	0.48	0.60
# Correct Answers	209	263
Precision	0.88	0.89
Recall	0.42	0.53
F-measure	0.57	0.66

Table 11. Performance evaluation of three systems

Method **Evaluation**	**TermMine QE+**	**Google Translate**	**Yahoo! Babelfish**
# of cases done	300	146	44
# of correct answers	263	67	23
Applicability	0.60	0.29	0.09
Precision	0.89	0.46	0.52
Recall	0.53	0.13	0.05
F-measure	0.66	0.21	0.08

6 Comparison with Previous Work

Machine transliteration has been an area of active research. Most of the machine transliteration method attempts to model the transliteration process of mapping between graphemes and phonemes. Knight and Graehl (1998) proposed a multilayer model and a generate-and-test approach to perform back transliteration from Japanese to English based on the model. In our work we address an issue of producing transliteration by way of search.

Goto et al. (2003), and Li et al. (2004) proposed a grapheme-based transliteration model. Hybrid transliteration models were described by Al-Onaizan and Knight (2002), and Oh et al. (2005).

Recently, some of the machine transliteration study has begun to consider the problem of extracting names and their transliterations from parallel corpora (Qu and Grefenstette 2004, Lin, Wu and Chang 2004; Lee and Chang 2003, Li and Grefenstette 2005).

Cao and Li (2002) described a new method for base noun phrase translation by using Web data. Kwok, et al. (2001) described a system called *CHINET* for cross language name search. Nagata et al. (2001) described how to exploit proximity and redundancy to extract translation for a given term. Lu, Chien, and Lee (2002) describe a method for name translation based on mining of anchor texts. More recently, Zhang, Huang, and Vogel (2005) proposed to use occurring words to expand queries for searching and extracting transliterations. Oh and Isahara (2006) use phonetic-similarity to recognize transliteration pairs on the Web.

In contrast to previous work, we propose a simple method for extracting transliterations based on a statistical model trained automatically on a bilingual name list via unsupervised learning. We also carried out experiments and evaluation of training and applying the proposed model to extract transliterations by using web as corpus.

7 Conclusion and Future Work

Morphological query expansion represents an innovative way to capture cross-language relations in name transliteration. The method is independent of the bilingual lexicon content making it easy to adopt to other proper names such person, product, or organization names. This approach is useful in a number of machine translation subtasks, including name transliteration, back transliteration, named entity translation, and terminology translation.

Many opportunities exist for future research and improvement of the proposed approach. First, the method explored here can be extended as an alterative way to support such MT subtasks as back transliteration (Knight and Graehl 1998) and noun phrase translation (Koehn and Knight 2003). Finally, for more challenging MT tasks, such as handling sentences, the improvement of translation quality probably will also be achieved by combining this IR-based approach and statistical machine translation. For example, a pre-processing unit may replace the proper names in a sentence with transliterations (e.g., mixed code text "*The cities of* 美索不達米亞 *prospered under* 巴底亞 *and* 薩珊 *rule*." before sending it off to MT for final translation.

References

GW Bian, HH Chen. Cross-language information access to multilingual collections on the internet. 2000. *Journal of American Society for Information Science & Technology (JASIST), Special Issue on Digital Libraries*, 51(3), pp.281-296, 2000.

Y. Cao and H. Li. Base Noun Phrase Translation Using Web Data and the EM Algorithm. 2002. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pp.127-133, 2002.

PJ. Cheng, JW. Teng, RC. Chen, JH. Wang, WH. Lu, and LF. Chien. Translating unknown queries with web corpora for cross-language information retrieval. 2004. In *Proceedings of the 27th ACM International Conference on Research and Development in Information Retrieval (SIGIR04)*, pp. 146-153, 2004.

I. Goto, N. Kato, N. Uratani, and T. Ehara. Transliteration considering context information based on the maximum entropy method. In *Proceedings of Ninth Machine Translation Summit*, pp.125-132, 2003.

F. Huang, S. Vogel, and A. Waibel. Automatic extraction of named entity translingual equivalence based on multi-feature cost minimization. In *Proceeding of the 41st ACL, Workshop on Multilingual and Mixed-Language Named Entity Recognition*, Sapporo, 2003.

A. Kilgarriff and Grefenstette, G. 2003. Introduction to the Special Issue on the Web as Corpus. *Computational Linguistics 29(3)*, pp. 333-348, 2003.

K. Knight, J. Graehl. Machine Transliteration. 1998. *Computational Linguistics 24(4)*, pp.599-612, 1998.

P. Koehn, K. Knight. 2003. Feature-Rich Statistical Translation of Noun Phrases. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pp. 311-318, 2003.

J. Kupiec. 1993. An Algorithm for Finding Noun Phrase Correspondences in Bilingual Corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 17-22, 1993.

KL Kwok. 2001. NTCIR-2 Chinese, Cross Language Retrieval Experiments Using PIRCS. In *Proceedings of NTCIR Workshop Meeting*, pp.111-118, 2001.

KL Kwok, P Deng, N Dinstl, HL Sun, W Xu, P Peng, and Doyon, J. 2005. CHINET: a Chinese name finder system for document triage. In *Proceedings of 2005*

International Conference on Intelligence Analysis, 2005.

C.J. Lee, and Jason S. Chang. 2003. Acquisition of English-Chinese Transliterated Word Pairs from Parallel-Aligned Texts using a Statistical Machine Transliteration Model, In *Proceedings of HLT-NAACL 2003 Workshop*, pp. 96-103, 2003.

H. Li, M. Zhang, and J. Su. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pp.159-166, 2004.

Y. Li, G. Grefenstette. 2005. Translating Chinese Romanized name into Chinese idiographic characters via corpus and web validation. In *Proceedings of CORIA 2005*, pp. 323-338, 2005.

T. Lin, J.C. Wu, and J. S. Chang. 2004. Extraction of Name and Transliteration in Monolingual and Parallel Corpora. In *Proceedings of AMTA 2004*, pp.177-186, 2004.

WH. Lu, LF. Chien, and HJ. Lee. 2002. Translation of web queries using anchor text mining. *ACM Transactions on Asian Language Information Processing, 1(2)*:159–172, 2002.

WH Lu, LF Chien, HJ Lee. Anchor text mining for translation of Web queries: A transitive translation approach. *ACM Transactions on Information Systems 22(2)*, pp. 242-269, 2004.

M. Nagata, T. Saito, and K. Suzuki. Using the Web as a bilingual dictionary. 2001. In *Proceedings of 39th. ACL Workshop on Data-Driven Methods in Machine Translation*, pp. 95-102, 2001.

J.-H Oh, and H. Isahara. 2006. Mining the Web for Transliteration Lexicons: Joint-Validation Approach, In *IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 254-261, 2006.

J.-H. Oh and K.-S. Choi. 2005. An ensemble of grapheme and phoneme for machine transliteration. In *Proceedings of IJCNLP05*, pp.450–461, 2005.

Y. Qu, and G. Grefenstette. 2004. Finding Ideographic Representations of Japanese Names Written in Latin Script via Language Identification and Corpus Validation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pp.183-190, 2004.

CK Quah. 2006. Translation and Technology, *Palgrave Textbooks in Translation and Interpretation*, Palgrave MacMillan.

Y Zhang, F Huang, S Vogel. 2005. Mining translations of OOV terms from the web through cross-lingual query expansion. In *Proceedings of the 28th Annual International ACM SIGIR*, pp.669-670, 2005.

Y. Zhang and P. Vines. 2004. Detection and translation of oov terms prior to query time. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp.524-525, 2004.

Learning to Merge Word Senses

Rion Snow **Sushant Prakash**
Computer Science Department
Stanford University
Stanford, CA 94305 USA
{rion,sprakash}@cs.stanford.edu

Daniel Jurafsky
Linguistics Department
Stanford University
Stanford, CA 94305 USA
jurafsky@stanford.edu

Andrew Y. Ng
Computer Science Department
Stanford University
Stanford, CA 94305 USA
ang@cs.stanford.edu

Abstract

It has been widely observed that different NLP applications require different sense granularities in order to best exploit word sense distinctions, and that for many applications WordNet senses are too fine-grained. In contrast to previously proposed automatic methods for sense clustering, we formulate sense merging as a supervised learning problem, exploiting human-labeled sense clusterings as training data. We train a discriminative classifier over a wide variety of features derived from WordNet structure, corpus-based evidence, and evidence from other lexical resources. Our learned similarity measure outperforms previously proposed automatic methods for sense clustering on the task of predicting human sense merging judgments, yielding an absolute F-score improvement of 4.1% on nouns, 13.6% on verbs, and 4.0% on adjectives. Finally, we propose a model for clustering sense taxonomies using the outputs of our classifier, and we make available several automatically sense-clustered WordNets of various sense granularities.

1 Introduction

Defining a discrete inventory of senses for a word is extremely difficult (Kilgarriff, 1997; Hanks, 2000; Palmer et al., 2005). Perhaps the greatest obstacle is the dynamic nature of sense definition: the correct granularity for word senses depends on the application. For language learners, a fine-grained set of word senses may help in learning subtle distinctions, while coarsely-defined senses are probably more useful in NLP tasks like information retrieval (Gonzalo et al., 1998), query expansion (Moldovan and Mihalcea, 2000), and WSD (Resnik and Yarowsky, 1999; Palmer et al., 2005).

Lexical resources such as WordNet (Fellbaum, 1998) use extremely fine-grained notions of word sense, which carefully capture even minor distinctions between different possible word senses (e.g., the 8 noun senses of *bass* shown in Figure 1). Producing sense-clustered inventories of arbitrary sense granularity is thus crucial for tasks which depend on lexical resources like WordNet, and is also important for the task of automatically constructing new WordNet-like taxonomies. A solution to this problem must also deal with the constraints of the WordNet taxonomy itself; for example when clustering two senses, we need to consider the transitive effects of merging synsets.

The state of the art in sense clustering is insufficient to meet these needs. Current sense clustering algorithms are generally unsupervised, each relying on a different set of useful features or hand-built rules. But hand-written rules have little flexibility to produce clusterings of different granularities, and previously proposed methods offer little in the direction of intelligently combining and weighting the many proposed features.

In response to these challenges, we propose a new algorithm for clustering large-scale sense hierarchies like WordNet. Our algorithm is based on a supervised classifier that learns to make graduated judgments corresponding to the estimated probability that each particular sense pair should be merged. This classifier is trained on gold standard sense clustering judgments using a diverse feature space. We are able to use the outputs of our classifier to produce a ranked list of sense merge judgments by merge probability, and from this create sense-clustered inventories of arbitrary sense granularity.[1]

In Section 2 we discuss past work in sense cluster-

[1] We have made sense-clustered Wordnets using the algorithms discussed in this paper available for download at **http://ai.stanford.edu/∼rion/swn**.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1005–1014, Prague, June 2007. ©2007 Association for Computational Linguistics

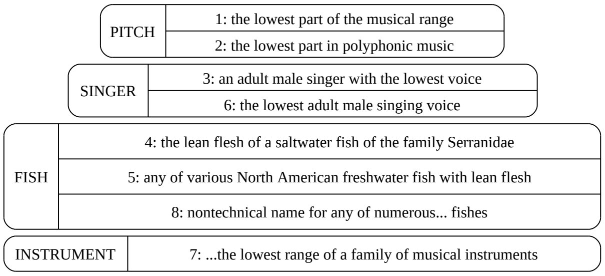

PITCH
1: the lowest part of the musical range
2: the lowest part in polyphonic music
SINGER
3: an adult male singer with the lowest voice
6: the lowest adult male singing voice
FISH
4: the lean flesh of a saltwater fish of the family Serranidae
5: any of various North American freshwater fish with lean flesh
8: nontechnical name for any of numerous... fishes
INSTRUMENT
7: ...the lowest range of a family of musical instruments

Figure 1: Sense clusters for the noun *bass*; the eight WordNet senses as clustered into four groups in the SENSEVAL-2 coarse-grained evaluation data

ing, and the gold standard datasets that we use in our work. In Section 3 we introduce our battery of features; in Section 4 we show how to extend our sense-merging model to cluster full taxonomies like WordNet. In Section 5 we evaluate our classifier against thirteen previously proposed methods.

2 Background

A wide number of manual and automatic techniques have been proposed for clustering sense inventories and mapping between sense inventories of different granularities. Much work has gone into methods for measuring synset similarity; early work in this direction includes (Dolan, 1994), which attempted to discover sense similarities between dictionary senses. A variety of synset similarity measures based on properties of WordNet itself have been proposed; nine such measures are discussed in (Pedersen et al., 2004), including gloss-based heuristics (Lesk, 1986; Banerjee and Pedersen, 2003), information-content based measures (Resnik, 1995; Lin, 1998; Jiang and Conrath, 1997), and others. Other approaches have used specific cues from WordNet structure to inform the construction of semantic rules; for example, (Peters et al., 1998) suggest clustering two senses based on a wide variety of structural cues from WordNet, including if they are *twins* (if two synsets share more than one word in their synonym list) or if they represent an example of *autohyponymy* (if one sense is the direct descendant of the other). (Mihalcea and Moldovan, 2001) implements six semantic rules, using *twin* and *autohyponym* features, in addition to other WordNet-structure-based rules such as whether two synsets share a *pertainym*, *antonym*, or are clustered together in the same *verb group*.

A large body of work has attempted to capture corpus-based estimates of word similarity (Pereira et al., 1993; Lin, 1998); however, the lack of large sense-tagged corpora prevent most such techniques from being used effectively to compare different senses of the same word. Some corpus-based attempts that are capable of estimating similarity between word senses include the *topic signatures* method; here, (Agirre and Lopez, 2003) collect contexts for a polysemous word based either on sense-tagged corpora or by using a weighted agglomeration of contexts of a polysemous word's monosemous relatives (i.e., single-sense synsets related by hypernym, hyponym, or other relations) from some large untagged corpus. Other corpus-based techniques developed specifically for sense clustering include (McCarthy, 2006), which uses a combination of word-to-word distributional similarity combined with the JCN WordNet-based similarity measure, and work by (Chugur et al., 2002) in finding co-occurrences of senses within documents in sense-tagged corpora. Other attempts have exploited disagreements between WSD systems (Agirre and Lopez, 2003) or between human labelers (Chklovski and Mihalcea, 2003) to create synset similarity measures; while promising, these techniques are severely limited by the performance of the WSD systems or the amount of available labeled data.

Some approaches for clustering have made use of regular patterns of polysemy among words. (Peters et al., 1998) uses the COUSIN relation defined in WordNet 1.5 to cluster hyponyms of categorically related noun synsets, e.g., "container/quantity" (e.g., for clustering senses of "cup" or "barrel") or "organization/construction" (e.g., for the building and institution senses of "hospital" or "school"); other approaches based on systematic polysemy include the hand-constructed CORELEX database (Buitelaar, 1998), and automatic attempts to extract patterns of systematic polysemy based on minimal description length principles (Tomuro, 2001).

Another family of approaches has been to use either manually-annotated or automatically-constructed mappings to coarser-grained sense inventories; an attempt at providing coarse-grained sense distinctions for the SENSEVAL-1 exercise included a mapping between WordNet and the Hector lexicon (Palmer et al., 2005). Other attempts in this vein include mappings between WordNet and PropBank (Palmer et al., 2004) and mappings to Levin classes (Levin, 1993; Palmer et al., 2005). (Navigli, 2006) presents an automatic approach for mapping between sense inventories; here similarities in gloss definition and structured relations between the two sense inventories are exploited in order to map between WordNet senses and distinctions made within the coarser-grained Oxford English Dictionary. Other work has attempted to exploit translational equivalences of WordNet senses in other languages, for example using foreign language WordNet interlingual indexes (Gonzalo et al., 1998; Chugur et al., 2002).

2.1 Gold standard sense clustering data

Our approach for learning how to merge senses relies upon the availability of labeled judgments of sense relatedness. In this work we focus on two datasets of hand-labeled sense groupings for WordNet: first, a dataset of sense groupings over nouns, verbs, and adjectives provided as part of the SENSEVAL-2 English lexical sample WSD task (Kilgarriff, 2001), and second, a corpus-driven mapping of nouns and verbs in WordNet 2.1 to the Omega Ontology (Philpot et al., 2005), produced as part of the ONTONOTES project (Hovy et al., 2006).

A wide variety of semantic and syntactic criteria were used to produce the SENSEVAL-2 groupings (Palmer et al., 2004; Palmer et al., 2005); this data covers all senses of 411 nouns, 519 verbs, and 257 adjectives, and has been used as gold standard sense clustering data in previous work (Agirre and Lopez, 2003; McCarthy, 2006)[2]. The number of judgments within this data (after mapping to WordNet 2.1) is displayed in Table 1.

Due to a lack of interannotator agreement data for this dataset, (McCarthy, 2006) performed an annotation study using three labelers on a 20-noun subset of the SENSEVAL-2 groupings; the three labelers were given the task of deciding whether the 351 potentially-related sense pairs were “Related”, “Unrelated”, or “Don’t Know”.[3] In this task the pairwise interannotator F-scores were (0.4874, 0.5454, 0.7926), for an average F-score of 0.6084.

SENSEVAL-2

POS	Total Pairs	Merged Pairs	Proportion
Nouns	16403	2593	0.1581
Verbs	30688	3373	0.1099
Adjectives	8368	2209	0.2640

ONTONOTES

POS	Total Pairs	Merged Pairs	Proportion
Nouns	3552	347	0.0977
Verbs	4663	1225	0.2627

Table 1: Gold standard datasets for sense merging; only sense pairs that share a word in common are included; proportion refers to the fraction of synsets sharing a word that have been merged

POS	Overlap	ON-True		ON-False		F-Score
		S-T	S-F	S-T	S-F	
Nouns	2116	121	55	181	1759	0.5063
Verbs	3297	351	503	179	2264	0.5072

Table 2: Agreement data for gold standard datasets

The ONTONOTES dataset[4] covers a smaller set of nouns and verbs, but it has been created with a more rigorous corpus-based iterative annotation process. For each of the nouns and verbs in question, a 50-sentence sample of instances is annotated using a preliminary set of sense distinctions; if the word sense interannotator agreement for the sample is less than 90%, then the sense distinctions are revised and the sample is re-annotated, and so forth, until an interannotator agreement of at least 90% is reached.

We construct a combined gold standard set from these SENSEVAL-2 and ONTONOTES groupings, removing disagreements. The overlap and agreement/disagreement data between the two groupings is given in Table 2; here, for example, the column with **ON-True** and **S-F** indicates the count of senses that ONTONOTES judged as positive examples of sense merging, but that SENSEVAL-2 data did not merge. We also calculate the F-score achieved by considering only one of the datasets as a gold standard, and computing precision and recall for the other. Since the two datasets were created independently, with different annotation guidelines, we can-

[2] In order to facilitate future work in this area, we have made cleaned versions of these groupings available at **http://ai.stanford.edu/∼rion/swn** along with a “diff” with the original files.

[3] McCarthy’s gold standard data is available at **ftp://ftp.informatics.susx.ac.uk/pub/users/dianam/relateGS/**.

[4] The OntoNotes groupings will be available through the LDC at **http://www.ldc.upenn.edu**.

not consider this as a valid estimate of interannotator agreement; nonetheless the F-score for the two datasets on the overlapping set of sense judgments (50.6% for nouns and 50.7% for verbs) is roughly in the same range as those observed in (McCarthy, 2006).

3 Learning to merge word senses

3.1 WordNet-based features

Here we describe the feature space we construct for classifying whether or not a pair of synsets should be merged; first, we employ a wide variety of linguistic features based on information derived from WordNet. We use eight similarity measures implemented within the WordNet::Similarity package[5], described in (Pedersen et al., 2004); these include three measures derived from the paths between the synsets in WordNet: HSO (Hirst and St-Onge, 1998), LCH (Leacock and Chodorow, 1998), and WUP (Wu and Palmer, 1994); three measures based on information content: RES (Resnik, 1995), LIN (Lin, 1998), and JCN (Jiang and Conrath, 1997); the gloss-based Extended Lesk Measure LESK, (Banerjee and Pedersen, 2003), and finally the gloss vector similarity measure VECTOR (Patwardan, 2003). We implement the TWIN feature (Peters et al., 1998), which counts the number of shared synonyms between the two synsets. Additionally we produce pairwise features indicating whether two senses share an ANTONYM, PERTAINYM, or derivationally-related forms (DERIV). We also create the verb-specific features of whether two verb synsets are linked in a VERBGROUP (indicating semantic similarity) or share a VERBFRAME, indicating syntactic similarity. Also, we encode a generalized notion of siblinghood in the MN features, recording the distance of the synset pair's nearest least common subsumer (i.e., closest shared hypernym) from the two synsets, and, separately, the maximum of those distances (in the MAXMN feature.

Previous attempts at categorizing systematic polysemy patterns within WordNet has resulted in the COUSIN feature[6]; we create binary features which indicate whether a synset pair belong to hypernym ancestries indicated by one or more of these COUSIN features, and the specific cousin pair(s) involved. Finally we create sense-specific features, including SENSECOUNT, the total number of senses associated with the shared word between the two synsets with the highest number of senses, and SENSENUM, the specific pairing of senses for the shared word with the highest number of senses (which might allow us to learn whether the most frequent sense of a word has a higher chance of having similar derivative senses with lower frequency).

3.2 Features derived from corpora and other lexical resources

In addition to WordNet-based features, we use a number of features derived from corpora and other lexical resources. We use the publicly available topic signature data[7] described in (Agirre and Lopez, 2004), yielding representative contexts for all nominal synsets from WordNet 1.6. These topic signatures were obtained by weighting the contexts of monosemous relatives of each noun synset (i.e., single-sense synsets related by hypernym, hyponym, or other relations); the text for these contexts were extracted from snippets using the Google search engine. We then create a sense similarity feature by taking a thresholded cosine similarity between pairs of topic signatures for these noun synsets.

Additionally, we use the WordNet domain dataset described in (Magnini and Cavaglia, 2000; Bentivogli et al., 2004). This dataset contains one or more labels indicating of 164 hierarchically organized "domains" or "subject fields" for each noun, verb, and adjective synset in WordNet; we derive a set of binary features from this data, with a single feature indicating whether or not two synsets share a domain, and one indicator feature per pair of domains indicating respective membership of the sense pair within those domains.

Finally, we use as a feature the mappings produced in (Navigli, 2006) of WordNet senses to Oxford English Dictionary senses. This OED dataset was used as the coarse-grained sense inventory in the Coarse-grained English all-words task of SemEval-

[5] We choose not to use the PATH measure due to its negligible difference from the LCH measure.

[6] This data is included in the WordNet 1.6 distribution as the "cousin.tops" file.

[7] The topic signature data is available for download at **http://ixa.si.ehu.es/Ixa/resources/sensecorpus**.

2007[8]; we specify a single binary feature for each pair of synsets from this data; this feature is true if the words are clustered in the OED mapping, and false otherwise.

3.3 Classifier, training, and feature selection

For each part of speech, we split the merged gold standard data into a part-of-speech-specific training set (70%) and a held-out test set (30%). For every synset pair we use the binary "merged" or "not-merged" labels to train a support vector machine classifier[9] (Joachims, 2002) for each POS-specific training set. We perform feature selection and regularization parameter optimization using 10-fold cross-validation.

4 Clustering Senses in WordNet

The previous section describes a classifier which predicts whether two synsets should be merged; we would like to use the pairwise judgments of this classifier to cluster the senses within a sense hierarchy. In this section we present the challenge implicit in applying sense merging to full taxonomies, and present our model for clustering within a taxonomy.

4.1 Challenges of clustering a sense taxonomy

The task of clustering a sense taxonomy presents certain challenges not present in the problem of clustering the senses of a word; in order to create a consistent clustering of a sense hierarchy an algorithm must consider the transitive effects of merging synsets. This problem is compounded in sense taxonomies like WordNet, where each synset may have additional structured relations, e.g., hypernym (IS-A) or holonym (is-part-of) links. In order to consistently merge two noun senses with different hypernym ancestries within WordNet, for example, an algorithm must decide whether to have the new sense inherit both hypernym ancestries, or whether to inherit only one, and if so it must decide which ancestry is more relevant for the merged sense.

Without strict checking, human labelers will likely find it difficult to label a sense inventory with transitively-consistent judgments. As an example, consider the SENSEVAL-2 clusterings of the verbs *require* and *need*, as shown in Figure 2. In WN 2.1 *require* has four verb senses, of which the first has synonyms {*necessitate*, *ask*, *postulate*, *need*, *take*, *involve*, *call for*, *demand*}, and gloss "require as useful, just, or proper"; and the fourth has synonyms {*want*, *need*}, and gloss "have need of."

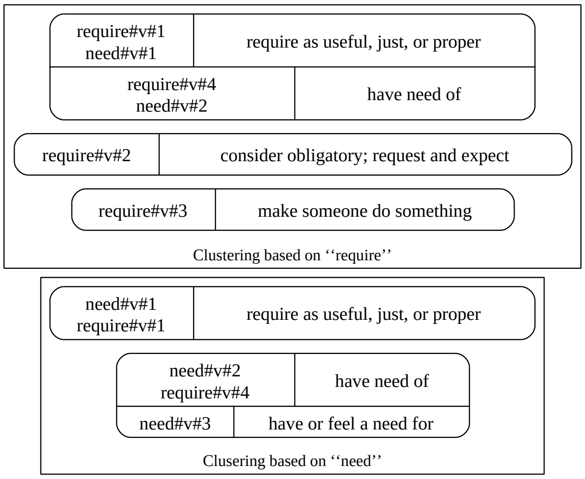

require#v#1
need#v#1
require as useful, just, or proper
require#v#4
need#v#2
have need of
require#v#2
consider obligatory; request and expect
require#v#3
make someone do something
Clustering based on ''require''
need#v#1
require#v#1
require as useful, just, or proper
need#v#2
require#v#4
have need of
need#v#3
have or feel a need for
Clusering based on ''need''

Figure 2: Inconsistent sense clusters for the verbs *require* and *need* from SENSEVAL-2 judgments

Within the word *require*, the SENSEVAL-2 dataset clusters senses 1 and 4, leaving the rest unclustered. In order to make a consistent clustering with respect to the sense inventory, however, we must enforce the transitive closure by merging the synset corresponding to the first sense (*necessitate*, *ask*, *need* etc.), with the senses of *want* and *need* in the fourth sense. In particular, these two senses correspond to WordNet 2.1 senses *need#v#1* and *need#v#2*, respectively, which are **not** clustered according to the SENSEVAL-2 word-specific labeling for *need* – *need#v#1* is listed as a singleton (i.e., unclustered) sense, though *need#v#2* is clustered with *need#v#3*, "have or feel a need for."

While one might hope that such disagreements between sense clusterings are rare, we found 178 such transitive closure disagreements in the SENSEVAL-2 data. The ONTONOTES data is much cleaner in this respect, most likely due to the stricter annotation standard (Hovy et al., 2006); we found only one transitive closure disagreement

[8] **http://lcl.di.uniroma1.it/coarse-grained-aw/index.html**

[9] We use the SVM^{perf} package, freely available for non-commercial use from **http://svmlight.joachims.org**; we use the default settings in v2.00, except for the regularization parameter (set in 10-fold cross-validation).

in the OntoNotes data, specifically WordNet 2.1 synsets (*head#n#2*, *lead#n#7*: "be in charge of") and (*head#n#3, lead#v#4*: "travel in front of") are clustered under *head* but not under *lead*.

4.2 Sense clustering within a taxonomy

As a solution to the previously mentioned challenges, in order to produce taxonomies of different sense granularities with consistent sense distinctions we propose to apply agglomerative clustering over all synsets in WordNet 2.1. While one might consider recalculating synset similarity features after each synset merge operation, depending on the feature set this could be prohibitively expensive; for our purposes we use average-link agglomerative clustering, in effect approximating the the pairwise similarity score between a given synset and a merged sense as the average of the similarity scores between the given synset and the clustered sense's component synsets. Further, for the purpose of sense clustering we assume a zero sense similarity score between synsets with no intersecting words.

Without exploiting additional hypernym or coordinate-term evidence, our algorithm does not distinguish between judgments about which hypernym ancestry or other structured relationships to keep or remove upon merging two synsets. In lieu of additional evidence, for our experiments we choose to retain only the hypernym ancestry of the sense with the highest frequency in SEMCOR, breaking frequency ties by choosing the first-listed sense in WordNet. We add every other relationship (meronyms, entailments, etc.) to the new merged sense (except in the rare case where adding a relation would cause a cycle in acyclic relations like hypernymy or holonymy, in which case we omit it). Using this clustering method we have produced several sense-clustered WordNets of varying sense granularity, which we evaluate in Section 5.3.

5 Evaluation

We evaluate our classifier in a comparison with thirteen previously proposed similarity measures and automatic methods for sense clustering. We conduct a feature ablation study to explore the relevance of the different features in our system. Finally, we evaluate the sense-clustered taxonomies we create on the problem of providing improved coarse-grained sense distinctions for WSD evaluation.

5.1 Evaluation of automatic sense merging

We evaluate our classifier on two held-out test sets; first, a 30% sample of the sense judgments from the merged gold standard dataset consisting of both the SENSEVAL-2 and ONTONOTES sense judgments; and, second, a test set consisting of only the ONTONOTES subset of our first held-out test set. For comparison we implement thirteen of the methods discussed in Section 2. First, we evaluate each of the eight WordNet::Similarity measures individually. Next, we implement cosine similarity of topic signatures (TOPSIG) built from monosemous relatives (Agirre and Lopez, 2003), which provides a real-valued similarity score for noun synset pairs.

Additionally, we implement the two methods proposed in (Peters et al., 1998), namely using metonymy clusters (MetClust) and generalization clusters (GenClust) based on the COUSIN relationship in WordNet. While (Peters et al., 1998) only considers four cousin pairs, we re-implement their method for general purpose sense clustering by using all 226 cousin pairs defined in WordNet 1.6, mapped to WordNet 2.1 synsets. These methods each provide a single clustering of noun synsets.

Next, we implement the set of semantic rules described in (Mihalcea and Moldovan, 2001) (MIMO); this algorithm for merging senses is based on 6 semantic rules, in effect using a subset of the TWIN, MAXMN, PERTAINYM, ANTONYM, and VERBGROUP features; in our implementation we set the parameter for when to cluster based on number of twins to $K = 2$; this results in a single clustering for each of nouns, verbs, and adjectives. Finally, we compare against the mapping from WordNet to the Oxford English Dictionary constructed in (Navigli, 2006), equivalent to clustering based solely on the OED feature.

Considering merging senses as a binary classification task, Table 3 gives the F-score performance of our classifier vs. the thirteen other classifiers and an uninformed "merge all synsets" baseline on our held-out gold standard test set. This table shows that our SVM classifier outperforms all implemented methods on the basis of F-score on both datasets

	SENSEVAL-2 + ONTONOTES			ONTONOTES	
Method	Nouns	Verbs	Adj	Nouns	Verbs
SVM	**0.4228**	**0.4319**	**0.4727**	**0.3698**	**0.4545**
RES	0.3817	0.2703	—	0.2807	0.3156
WUP	0.3763	0.2782	—	0.3036	0.3451
LCH	0.3700	0.2440	—	0.2857	0.3396
OED	0.3310	0.2878	0.3712	0.2183	0.3962
LESK	0.3174	0.2956	0.4323	0.2914	0.3774
HSO	0.3090	0.2784	0.4312	0.3025	0.3156
TOPSIG	0.3072	—	—	0.2581	—
VEC	0.2960	0.2315	0.4321	0.2454	0.3420
JCN	0.2818	0.2292	—	0.2222	0.3156
LIN	0.2759	0.2464	—	0.2056	0.3471
Baseline	0.2587	0.2072	0.4312	0.1488	0.3156
MIMO	0.0989	0.2142	0.0759	0.1833	0.2157
GenClust	0.0973	—	—	0.0264	—
MetClust	0.0876	—	—	0.0377	—

Table 3: F-score sense merging evaluation on hand-labeled testsets

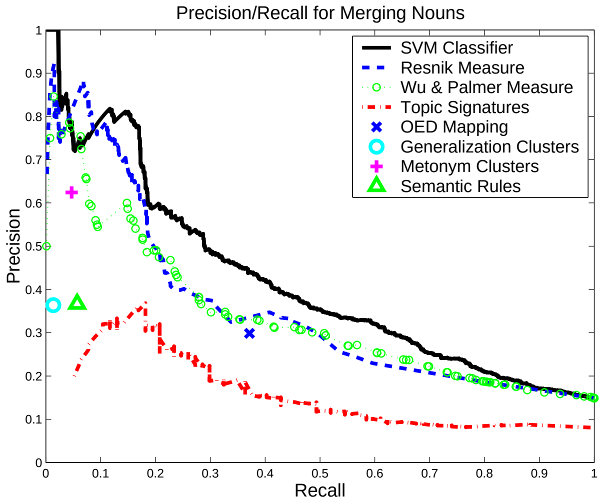


Figure 3: Precision/Recall plot for noun sense merge judgments

for all parts of speech. In Figure 3 we give a precision/recall plot for noun sense merge judgments for the SENSEVAL-2 + ONTONOTES dataset. For sake of simplicity we plot only the two best measures (RES and WUP) of the eight WordNet-based similarity measures; we see that our classifier, RES, and WUP each have higher precision all levels of recall compared to the other tested measures.

Of the methods we compare against, only the WordNet-based similarity measures, (Mihalcea and Moldovan, 2001), and (Navigli, 2006) provide a method for predicting verb similarities; our learned measure widely outperforms these methods, achieving a 13.6% F-score improvement over the LESK similarity measure. In Figure 4 we give a precision/recall plot for verb sense merge judgments, plotting the performance of the three best WordNet-based similarity measures; here we see that our classifier has significantly higher precision than all other tested measures at nearly every level of recall.

Only the measures provided by LESK, HSO, VEC, (Mihalcea and Moldovan, 2001), and (Navigli, 2006) provide a method for predicting adjective similarities; of these, only LESK and VEC outperform the uninformed baseline on adjectives, while our learned measure achieves a 4.0% improvement over the LESK measure on adjectives.

5.2 Feature analysis

Next we analyze our feature space. Table 4 gives the ablation analysis for all features used in our system as evaluated on our held-out test set; here the quantity listed in the table is the F-score loss obtained by removing that single feature from our feature space, and retraining and retesting our classifiers, keeping everything else the same. Here negative scores correspond to an *improvement* in classifier performance with the removal of the feature.

For noun classification, the three features that yield the highest gain in testset F-score are the topic signature, OED, and derivational link features, yielding a 4.0%, 3.6%, and 3.5% gain, respectively.

For verb classification, we find that three features yield more than a 5% F-score gain; by far the largest single-feature performance gain for verb classification found in our ablation study was the DERIV feature, i.e., the count of shared derivational links between the two synsets; this single feature improves our maximum F-score by 9.8% on the testset. This is a particularly interesting discovery, as none of the referenced automatic techniques for sense clustering presently make use of this very useful feature. We also achieve large gains with the LIN and LESK similarity features, with F-score improvement of 7.4% and 5.4% gain respectively.

For adjective classification again the DERIV feature proved very helpful, with a 3.5% gain on the testset. Interestingly, only the DERIV feature and the SENSECNT features helped across all parts of speech; in many cases a feature which proved to be

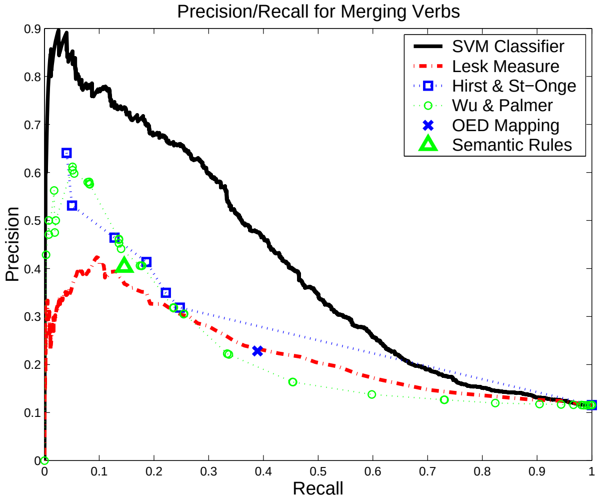


Figure 4: Precision/Recall plot for verb sense merge judgments

very helpful for one part of speech actually hurt performance on another part of speech (e.g., LIN on nouns and OED on adjectives).

5.3 Evaluation of sense-clustered Wordnets

Our goal in clustering a sense taxonomy is to produce fully sense-clustered WordNets, and to be able to produce coarse-grained Wordnets at many different levels of resolution. In order to evaluate the entire sense-clustered taxonomy, we have employed an evaluation method inspired by Word Sense Disambiguation (this is similar to an evaluation used in Navigli, 2006, however we do not remove monosemous clusters). Given past system responses in the SENSEVAL-3 English all-words task, we can evaluate past systems on the same corpus, but using the coarse-grained sense hierarchy provided by our sense-clustered taxonomy. We may then compare the scores of each system on the coarse-grained task against their scores given a random clustering at the same resolution. Our expectation is that, if our sense clustering is much better than a random sense clustering (and, of course, that the WSD algorithms perform better than random guessing), we will see a marked improvement in the performance of WSD algorithms using our coarse-grained sense hierarchy.

We consider the outputs of the top 3 all-words WSD systems that participated in Senseval-3: Gambl (Decadt et al., 2004), SenseLearner (Mihalcea and Faruque, 2004), and KOC University (Yuret, 2004). A guess by a system is given full credit if it was either the correct answer or if it was in the same cluster as the correct answer.

	Nouns	Verbs	Adjectives
F-SCORE	0.4228	0.4319	0.4727
Feature	F-Score Ablation Difference		
TOPSIG	0.0403	—	—
OED	0.0355	0.0126	-0.0124
DERIV	0.0351	0.0977	0.0352
RES	0.0287	0.0147	—
TWIN	0.0285	0.0109	-0.0130
MN	0.0188	0.0358	—
LESK	0.0183	0.0541	-0.0250
SENSENUM	0.0155	0.0146	-0.0147
SENSECNT	0.0121	0.0160	0.0168
DOMAIN	0.0119	0.0082	-0.0265
LCH	0.0099	0.0068	—
WUP	0.0036	0.0168	—
JCN	0.0025	0.0190	—
ANTONYM	0.0000	0.0295	0.0000
MAXMN	-0.0013	0.0179	—
VEC	-0.0024	0.0371	-0.0062
HSO	-0.0073	0.0112	-0.0246
LIN	-0.0086	0.0742	—
COUSIN	-0.0094	—	—
VERBGRP	—	0.0327	—
VERBFRM	—	0.0102	—
PERTAINYM	—	—	-0.0029

Table 4: Feature ablation study; F-score difference obtained by removal of the single feature

Clearly any amount of clustering will only increase WSD performance. Therefore, to account for this natural improvement and consider only the effect of our particular clustering, we also calculate the expected score for a random clustering of the same granularity, as follows: Let C represent the set of clusters over the possible N synsets containing a given word; we then calculate the expectation that an incorrectly-chosen sense and the actual correct sense would be clustered together in the random clustering as $\frac{\sum_{c\in C}|c|(|c|-1)}{N(N-1)}$.

Our sense clustering algorithm provides little improvement over random clustering when too few or too many clusters are chosen; however, with an appropriate threshold for average-link clustering we find a maximum of 3.55% F-score improvement in WSD over random clustering (averaged over the decisions of the top 3 WSD algorithms). Table 5 shows the improvement of the three top WSD algorithms given a sense clustering created by our algorithm vs. a random clustering at the same granularity.

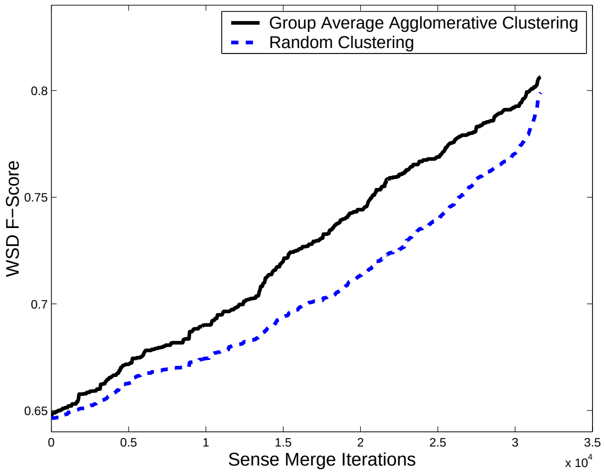


Figure 5: WSD Improvement with coarse-grained sense hierarchies

System	F-score	Avg-link	Random	Impr.
Gambl	0.6516	0.7702	0.7346	0.0356
SenseLearner	0.6458	0.7536	0.7195	0.0341
KOC Univ.	0.6414	0.7521	0.7153	0.0368

Table 5: Improvement in SENSEVAL-3 WSD performance using our average-link agglomerative clustering vs. random clustering at the same granularity

6 Conclusion

We have presented a classifier for automatic sense merging that significantly outperforms previously proposed automatic methods. In addition to its novel use of supervised learning and the integration of many previously proposed features, it is interesting that one of our new features, the DERIV count of shared derivational links between two synsets, proved an extraordinarily useful new cue for sense-merging, particularly for verbs.

We also show how to integrate this sense-merging algorithm into a model for sense clustering full sense taxonomies like WordNet, incorporating taxonomic constraints such as the transitive effects of merging synsets. Using this model, we have produced several WordNet taxonomies of various sense granularities; we hope these new lexical resources will be useful for NLP applications that require a coarser-grained sense hierarchy than that already found in WordNet.

Acknowledgments

Thanks to Marie-Catherine de Marneffe, Mona Diab, Christiane Fellbaum, Thad Hughes, and Benjamin Packer for useful discussions. Rion Snow is supported by an NSF Fellowship. This work was supported in part by the Disruptive Technology Office (DTO)'s Advanced Question Answering for Intelligence (AQUAINT) Phase III Program.

References

Eneko Agirre and Oier Lopez de Lacalle. 2003. Clustering WordNet word senses. In *Proceedings of RANLP 2003*.

Eneko Agirre and Oier Lopez de Lacalle. 2004. Publicly available topic signatures for all WordNet nominal senses. In *Proceedings of LREC 2004*.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended Gloss Overlaps as a Measure of Semantic Relatedness. In *Proceedings of IJCAI 2003*.

Lisa Bentivogli, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. 2004. Revising the WordNet Domains Hierarchy: Semantics, Coverage, and Balancing. In *Proceedings of COLING Workshop on Multilingual Linguistic Resources*, 2004.

Timothy Chklovski and Rada Mihalcea. 2003. Exploiting Agreement and Disagreement of Human Annotators for Word Sense Disambiguation. In *Proceedings of RANLP 2003*.

Irina Chugur, Julio Gonzalo, and Felisa Verdejo. 2002. Polysemy and Sense Proximity in the Senseval-2 Test Suite. In *Proceedings of ACL 2002 WSD Workshop*.

Bart Decadt, Veronique Hoste, Walter Daelemans, and Antal van den Bosch. 2004. Gamble, genetic algorithm optimization of memory-based wsd. In *Proceedings of ACL/SIGLEX Senseval-3*.

William Dolan. 1994. Word Sense Ambiguation: Clustering Related Senses. In *Proceedings of ACL 1994*.

Christiane Fellbaum. 1998. WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

Julio Gonzalo, Felia Verdejo, Irina Chugur, and Juan Cigarran. 1998. Indexing with WordNet synsets can improve text retrieval. In *Proceedings of COLING-ACL 1998 Workshop on WordNet in NLP Systems*.

Patrick Hanks. 2000. Do word meanings exist? *Computers and the Humanities*, **34**(1-2): 171-177.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An Electronic Lexical Database*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. *Proceedings of HLT-NAACL 2006*.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, 19-33.

Thorsten Joachims. 2002. Learning to Classify Text Using Support Vector Machines. Dissertation, Kluwer, 2002.

Adam Kilgariff. 1997. I don't believe in word senses. *Computers and the Humanities*, **31**(1-2): 1-13.

Adam Kilgarriff. 2001. English lexical sample task description. In *Proceedings of the* SENSEVAL-2 *workshop*, 17-20.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. In *WordNet: An Electronic Lexical Database*.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC 1986*.

Beth Levin. 1993. English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press, Chicago, IL.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of ICML 1998*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL 1998*.

Bernardo Magnini and Gabriela Cavaglia. 2000. Integrating Subject Field Codes into WordNet. In *Proceedings of LREC 2000*.

Diana McCarthy. 2006. Relating WordNet Senses for Word Sense Disambiguation. In *Proceedings of ACL Workshop on Making Sense of Sense, 2006*.

Rada Mihalcea and Dan I. Moldovan. 2001. Automatic Generation of a Coarse Grained WordNet. In *Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*.

Rada Mihalcea and Ehsanul Faruque. 2004. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In *Proceedings of ACL/SIGLEX Senseval-3*.

Dan I. Moldovan and Rada Mihalcea. 2000. Using WordNet and lexical operators to improve Internet searches. IEEE Internet Computing, 4(1):34-43.

Roberto Navigli. 2006. Meaningful Clustering of Senses Helps Boost Word Sense Disambiguation Performance. In *Proceedings of COLING-ACL 2006*.

Martha Palmer, Olga Babko-Malaya, Hoa Trang Dang. 2004. Different Sense Granularities for Different Applications. In *Proceedings of Workshop on Scalable Natural Language Understanding*.

Martha Palmer, Hoa Trang Dang, and Christiane Fellbaum. 2005. Making fine-grained and coarse-grained sense distinctions. Journal of Natural Language Engineering.

Siddharth Patwardhan. 2003. Incorporating dictionary and corpus information into a context vector measure of semantic relatedness. Master's thesis, Univ. of Minnesota, Duluth.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of NAACL 2004*.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional Clustering of English Words. In *Proceedings of ACL 1993*.

Wim Peters, Ivonne Peters, and Piek Vossen. 1998. Automatic Sense Clustering in EuroWordNet. In *Proceedings of LREC 1998.*

Andrew Philpot, Eduard Hovy, and Patrick Pantel. 2005. The Omega Ontology. In *Proceedings of the ONTOLEX Workshop at IJCNLP 2005*.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the IJCAI 1995*, 448-453.

Philip Resnik and David Yarowsky. 1999. Distinguishing systems and distinguishing senses: new evaluation methods for word sense disambiguation. Natural Language Engineering, 5(2):113-134.

Noriko Tomuro. 2001. Tree-cut and A Lexicon based on Systematic Polysemy. In *Proceedings of NAACL 2001*.

Zhibiao Wu and Martha Palmer. 1994. Verb Semantics and Lexical Selection. In *Proceedings of ACL 1994*.

Deniz Yuret. 2004. Some experiments with a naive bayes wsd system. In *Proceedings of ACL/SIGLEX Senseval-3*.

Improving Word Sense Disambiguation Using Topic Features

Jun Fu Cai, Wee Sun Lee
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{caijunfu, leews}@comp.nus.edu.sg

Yee Whye Teh
Gatsby Computational Neuroscience Unit
University College London
17 Queen Square, London WC1N 3AR, UK
ywteh@gatsby.ucl.ac.uk

Abstract

This paper presents a novel approach for exploiting the global context for the task of word sense disambiguation (WSD). This is done by using topic features constructed using the latent dirichlet allocation (LDA) algorithm on unlabeled data. The features are incorporated into a modified naïve Bayes network alongside other features such as part-of-speech of neighboring words, single words in the surrounding context, local collocations, and syntactic patterns. In both the English all-words task and the English lexical sample task, the method achieved significant improvement over the simple naïve Bayes classifier and higher accuracy than the best official scores on Senseval-3 for both task.

1 Introduction

Natural language tends to be ambiguous. A word often has more than one meanings depending on the context. Word sense disambiguation (WSD) is a natural language processing (NLP) task in which the correct meaning (sense) of a word in a given context is to be determined.

Supervised corpus-based approach has been the most successful in WSD to date. In such an approach, a corpus in which ambiguous words have been annotated with correct senses is first collected. Knowledge sources, or features, from the context of the annotated word are extracted to form the training data. A learning algorithm, like the support vector machine (SVM) or naïve Bayes, is then applied on the training data to learn the model. Finally, in testing, the learnt model is applied on the test data to assign the correct sense to any ambiguous word.

The features used in these systems usually include local features, such as part-of-speech (POS) of neighboring words, local collocations , syntactic patterns and global features such as single words in the surrounding context (bag-of-words) (Lee and Ng, 2002). However, due to the data scarcity problem, these features are usually very sparse in the training data. There are, on average, 11 and 28 training cases per sense in Senseval 2 and 3 lexical sample task respectively, and 6.5 training cases per sense in the SemCor corpus. This problem is especially prominent for the bag-of-words feature; more than hundreds of bag-of-words are usually extracted for each training instance and each feature could be drawn from any English word. A direct consequence is that the global context information, which the bag-of-words feature is supposed to capture, may be poorly represented.

Our approach tries to address this problem by clustering features to relieve the scarcity problem, specifically on the bag-of-words feature. In the process, we construct topic features, trained using the latent dirichlet allocation (LDA) algorithm. We train the topic model (Blei et al., 2003) on unlabeled data, clustering the words occurring in the corpus to a predefined number of topics. We then use the resulting topic model to tag the bag-of-words in the labeled corpus with topic distributions. We incorporate the distributions, called the topic features, using a simple Bayesian network, modified from naïve Bayes

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1015–1023, Prague, June 2007. ©2007 Association for Computational Linguistics

model, alongside other features and train the model on the labeled corpus. The approach gives good performance on both the lexical sample and all-words tasks on Senseval data.

The paper makes mainly two contributions. First, we are able to show that a feature that efficiently captures the global context information using LDA algorithm can significantly improve the WSD accuracy. Second, we are able to obtain this feature from unlabeled data, which spares us from any manual labeling work. We also showcase the potential strength of Bayesian network in the WSD task, obtaining performance that rivals state-of-arts methods.

2 Related Work

Many WSD systems try to tackle the data scarcity problem. Unsupervised learning is introduced primarily to deal with the problem, but with limited success (Snyder and Palmer, 2004). In another approach, the learning algorithm borrows training instances from other senses and effectively increases the training data size. In (Kohomban and Lee, 2005), the classifier is trained using grouped senses for verbs and nouns according to WordNet top-level synsets and thus effectively pooling training cases across senses within the same synset. Similarly, (Ando, 2006) exploits data from related tasks, using all labeled examples irrespective of target words for learning each sense using the Alternating Structure Optimization (ASO) algorithm (Ando and Zhang, 2005a; Ando and Zhang, 2005b). Parallel texts is proposed in (Resnik and Yarowsky, 1997) as potential training data and (Chan and Ng, 2005) has shown that using automatically gathered parallel texts for nouns could significantly increase WSD accuracy, when tested on Senseval-2 English all-words task.

Our approach is somewhat similar to that of using generic language features such as POS tags; the words are tagged with its semantic topic that may be trained from other corpuses.

3 Feature Construction

We first present the latent dirichlet allocation algorithm and its inference procedures, adapted from the original paper (Blei et al., 2003).

3.1 Latent Dirichlet Allocation

LDA is a probabilistic model for collections of discrete data and has been used in document modeling and text classification. It can be represented as a three level hierarchical Bayesian model, shown graphically in Figure 1. Given a corpus consisting of M documents, LDA models each document using a mixture over K topics, which are in turn characterized as distributions over words.

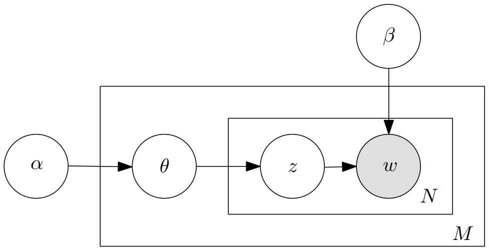


Figure 1: Graphical Model for LDA

In the generative process of LDA, for each document d we first draw the mixing proportion over topics θ_d from a Dirichlet prior with parameters α. Next, for each of the N_d words w_{dn} in document d, a topic z_{dn} is first drawn from a multinomial distribution with parameters θ_d. Finally w_{dn} is drawn from the topic specific distribution over words. The probability of a word token w taking on value i given that topic $z = j$ was chosen is parameterized using a matrix β with $\beta_{ij} = p(w = i|z = j)$. Integrating out θ_d's and z_{dn}'s, the probability $p(D|\alpha, \beta)$ of the corpus is thus:

$$\prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \right) d\theta_d$$

3.1.1 Inference

Unfortunately, it is intractable to directly solve the posterior distribution of the hidden variables given a document, namely $p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)$. However, (Blei et al., 2003) has shown that by introducing a set of variational parameters, γ and ϕ, a tight lower bound on the log likelihood of the probability can be found using the following optimization procedure:

$$(\gamma^*, \phi^*) = \arg\min_{\gamma,\phi} D(q(\theta, \mathbf{z}|\gamma, \phi) \| p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta))$$

where

$$q(\theta, \mathbf{z}|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n),$$

γ is the Dirichlet parameter for ϕ and the multinomial parameters $(\phi_1 \cdots \phi_N)$ are the free variational parameters. Note here γ is document specific instead of corpus specific like α. Graphically, it is represented as Figure 2. The optimizing values of γ and ϕ can be found by minimizing the Kullback-Leibler (KL) divergence between the variational distribution and the true posterior.

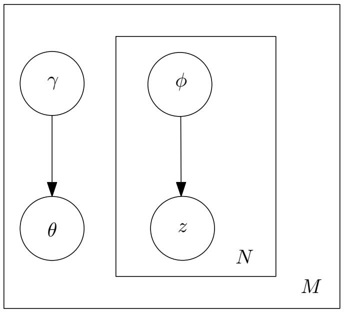


Figure 2: Graphical Model for Variational Inference

3.2 Baseline Features

For both the lexical sample and all-words tasks, we use the following standard *baseline features* for comparison.

POS Tags For each training or testing word, w, we include POS tags for P words prior to as well as after w within the same sentence boundary. We also include the POS tag of w. If there are fewer than P words prior or after w in the same sentence, we denote the corresponding feature as NIL.

Local Collocations Collocation $C_{i,j}$ refers to the ordered sequence of tokens (words or punctuations) surrounding w. The starting and ending position of the sequence are denoted i and j respectively, where a negative value refers to the token position prior to w. We adopt the same 11 collocation features as (Lee and Ng, 2002), namely $C_{-1,-1}$, $C_{1,1}$, $C_{-2,-2}$, $C_{2,2}$, $C_{-2,-1}$, $C_{-1,1}$, $C_{1,2}$, $C_{-3,-1}$, $C_{-2,1}$, $C_{-1,2}$, and $C_{1,3}$.

Bag-of-Words For each training or testing word, w, we get G words prior to as well as after w, within the same document. These features are position insensitive. The words we extract are converted back to their morphological root forms.

Syntactic Relations We adopt the same syntactic relations as (Lee and Ng, 2002). For easy reference, we summarize the features into Table 1.

POS of w	Features
Noun	Parent headword h POS of h Relative position of h to w
Verb	Left nearest child word of w, l Right nearest child word of w, r POS of l POS of r POS of w Voice of w
Adjective	Parent headword h POS of h

Table 1: Syntactic Relations Features

The exact values of P and G for each task are set according to cross validation result.

3.3 Topic Features

We first select an unlabeled corpus, such as 20 Newsgroups, and extract individual words from it (excluding stopwords). We choose the number of topics, K, for the unlabeled corpus and we apply the LDA algorithm to obtain the β parameters, where β represents the probability of a word w_i given a topic z_j, $p(w_i|z_j) = \beta_{ij}$. The model essentially clusters words that occurred in the unlabeled corpus according to K topics. The conditional probability $p(w_i|z_j) = \beta_{ij}$ is later used to tag the words in the unseen test example with the probability of each topic.

For some variants of the classifiers that we construct, we also use the γ parameter, which is document specific. For these classifiers, we may need to run the inference algorithm on the labeled corpus and possibly on the test documents. The γ parameter provides an approximation to the probability of selecting topic i in the document:

$$p(z_i|\gamma) = \frac{\gamma_i}{\sum_K \gamma_k}. \tag{1}$$

4 Classifier Construction

4.1 Bayesian Network

We construct a variant of the naïve Bayes network as shown in Figure 3. Here, w refers to the word. s refers to the sense of the word. In training, s is observed while in testing, it is not. The features f_1 to f_n are baseline features mentioned in Section 3.2 (including bag-of-words) while z refers to the latent topic that we set for clustering unlabeled corpus. The bag-of-words b are extracted from the neighbours of w and there are L of them. Note that L can be different from G, which is the number of bag-of-words in baseline features. Both will be determined by the validation result.

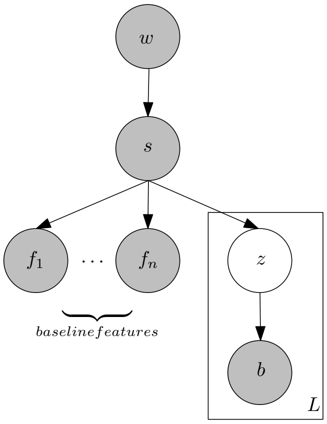


Figure 3: Graphical Model with LDA feature

The log-likelihood of an instance, $\ell(w, s, F, b)$ where F denotes the set of baseline features, can be written as

$$= \log p(w) + \log p(s|w) + \sum_F \log(p(f|s))$$
$$+ \sum_L \log\left(\sum_K p(z_k|s)p(b_l|z_k)\right).$$

The $\log p(w)$ term is constant and thus can be ignored. The first portion is normal naïve Bayes. And second portion represents the additional LDA plate. We decouple the training process into three separate stages. We first extract baseline features from the task training data, and estimate, using normal naïve Bayes, $p(s|w)$ and $p(f|s)$ for all w, s and f. The parameters associated with $p(b|z)$ are estimated using LDA from unlabeled data. Finally we estimate the parameters associated with $p(z|s)$. We experimented with three different ways of both doing the estimation as well as using the resulting model and chose one which performed best empirically.

4.1.1 Expectation Maximization Approach

For $p(z|s)$, a reasonable estimation method is to use maximum likelihood estimation. This can be done using the expectation maximization (EM) algorithm. In classification, we just choose s^* that maximizes the log-likelihood of the test instance, where:

$$s^* = \arg\max_s \ell(w, s, F, b)$$

In this approach, γ is never used which means the LDA inference procedure is not used on any labeled data at all.

4.1.2 Soft Tagging Approach

Classification in this approach is done using the full Bayesian network just as in the EM approach. However we do the estimation of $p(z|s)$ differently. Essentially, we perform LDA inference on the training corpus in order to obtain γ for each document. We then use the γ and β to obtain $p(z|b)$ for each word using

$$p(z_i|b_l, \gamma) = \frac{p(b_l|z_i)p(z_i|\gamma)}{\sum_K p(b_l|z_k)p(z_k|\gamma)},$$

where equation [1] is used for estimation of $p(z_i|\gamma)$.

This effectively transforms b to a topical distribution which we call a soft tag where each soft tag is probability distribution $t_1, \ldots, t_K$ on topics. We then use this topical distribution for estimating $p(z|s)$. Let s^i be the observed sense of instance i and $t_1^{ij}, \ldots, t_K^{ij}$ be the soft tag of the j-th bag-of-word feature of instance i. We estimate $p(z|s)$ as

$$p(z_{jk}|s) = \frac{\sum_{s^i=s} t_k^{ij}}{\sum_{s^i=s} \sum_{k'} t_{k'}^{ij}} \tag{2}$$

This approach requires us to do LDA inference on the corpus formed by the labeled training data, but not the testing data. This is because we need γ to get transformed topical distribution in order to learn $p(z|s)$ in the training. In the testing, we only apply the learnt parameters to the model.

4.1.3 Hard Tagging Approach

Hard tagging approach no longer assumes that z is latent. After $p(z|b)$ is obtained using the same procedure in Section 4.1.2, the topic z_i with the highest $p(z_i|b)$ among all K topics is picked to represent z. In this way, b is transformed into a single most "prominent" topic. This topic label is used in the same way as baseline features for both training and testing in a simple naïve Bayes model.

This approach requires us to perform the transformation both on the training as well as testing data, since z becomes an observed variable. LDA inference is done on two corpora, one formed by the training data and the other by testing data, in order to get the respective values of γ.

4.2 Support Vector Machine Approach

In the SVM (Vapnik, 1995) approach, we first form a training and a testing file using all standard features for each sense following (Lee and Ng, 2002) (one classifier per sense). To incorporate LDA feature, we use the same approach as Section 4.1.2 to transform b into soft tags, $p(z|b)$. As SVM deals with only observed features, we need to transform b both in the training data and in the testing data. Compared to (Lee and Ng, 2002), the only difference is that for each training and testing case, we have additional $L * K$ LDA features, since there are L bag-of-words and each has a topic distribution represented by K values.

5 Experimental Setup

We describe here the experimental setup on the English lexical sample task and all-words task.

We use MXPOST tagger (Adwait, 1996) for POS tagging, Charniak parser (Charniak, 2000) for extracting syntactic relations, SVMlight[1] for SVM classifier and David Blei's version of LDA[2] for LDA training and inference. All default parameters are used unless mentioned otherwise. For all standard baseline features, we use Laplace smoothing but for the soft tag (equation [2]), we use a smoothing parameter value of 2.

5.1 Development Process

5.1.1 Lexical Sample Task

We use the Senseval-2 lexical sample task for preliminary investigation of different algorithms, datasets and other parameters. As the dataset is used extensively for this purpose, only the Senseval-3 lexical sample task is used for evaluation.

Selecting Bayesian Network The best achievable result, using the three different Bayesian network approaches, when validating on Senseval-2 test data is shown in Table 2. The parameters that are used are P = 3 and G = 3.

EM	68.0
Hard Tagging	65.6
Soft Tagging	68.9

Table 2: Results on Senseval-2 English lexical sample using different Bayesian network approaches.

From the results, it appears that both the EM and the Hard Tagging approaches did not yield as good results as the Soft Tagging approach did. The EM approach ignores the LDA inference result, γ, which we use to get our topic prior. This information is document specific and can be regarded as global context information. The Hard Tagging approach also uses less information, as the original topic distribution is now represented only by the topic with the highest probability of occurring. Therefore, both methods have information loss and are disadvantaged against the Soft Tagging approach. We use the Soft Tagging approach for the Senseval-3 lexical sample and the all-words tasks.

Unlabeled Corpus Selection The unlabeled corpus we choose to train LDA include 20 Newsgroups, Reuters, SemCor, Senseval-2 lexical sample data and Senseval-3 lexical sample data. Although the last three are labeled corpora, we only need the words from these corpora and thus they can be regarded as unlabeled too. For Senseval-2 and Senseval-3 data, we define the whole passage for each training and testing instance as one document.

[1] http://svmlight.joachims.org

[2] http://www.cs.princeton.edu/~blei/lda-c/

The relative effect using different corpus and combinations of them is shown in Table 3, when validating on Senseval-2 test data using the Soft Tagging approach.

Corpus	$\|w\|$	K	L	Senseval-2
20 Newsgroups	1.7M	40	60	67.9
Reuters	1.3M	30	60	65.5
SemCor	0.3M	30	60	66.9
Senseval-2	0.6M	30	40	66.9
Senseval-3	0.6M	50	60	67.6
All	4.5M	60	40	68.9

Table 3: Effect of using different corpus for LDA training, $|w|$ represents the corpus size in terms of the number of words in the corpus

The 20 Newsgroups corpus yields the best result if used individually. It has a relatively larger corpus size at 1.7 million words in total and also a well balanced topic distribution among its documents, ranging across politics, finance, science, computing, etc. The Reuters corpus, on the other hand, focuses heavily on finance related articles and has a rather skewed topic distribution. This probably contributed to its inferior result. However, we found that the best result comes from combining all the corpora together with $K = 60$ and $L = 40$.

Results for Optimized Configuration As baseline for the Bayesian network approaches, we use naïve Bayes with all baseline features. For the baseline SVM approach, we choose P = 3 and include all the words occurring in the training and testing passage as bag-of-words feature.

The F-measure result we achieve on Senseval-2 test data is shown in Table 4. Our four systems are listed as the top four entries in the table. Soft Tag refers to the soft tagging Bayesian network approach. Note that we used the Senseval-2 test data for optimizing the configuration (as is done in the ASO result). Hence, the result should not be taken as reliable. Nevertheless, it is worth noting that the improvement of Bayesian network approach over its baseline is very significant ($+5.5\%$). On the other hand, SVM with topic features shows limited improvement over its baseline ($+0.8\%$).

Bayes (Soft Tag)	68.9
SVM-Topic	66.0
SVM baseline	65.2
NB baseline	63.4
ASO(best configuration)(Ando, 2006)	68.1
Classifier Combination(Florian, 2002)	66.5
Polynomial KPCA(Wu et al., 2004)	65.8
SVM(Lee and Ng, 2002)	65.4
Senseval-2 Best System	64.2

Table 4: Results (best configuration) compared to previous best systems on Senseval-2 English lexical sample task.

5.1.2 All-words Task

In the all-words task, no official training data is provided with Senseval. We follow the common practice of using the SemCor corpus as our training data. However, we did not use SVM approach in this task as there are too few training instances per sense for SVM to achieve a reasonably good accuracy.

As there are more training instances in SemCor, $230,000$ in total, we obtain the optimal configuration using 10 fold cross validation on the SemCor training data. With the optimal configuration, we test our system on both Senseval-2 and Senseval-3 official test data.

For baseline features, we set P = 3 and B = 1. We choose a LDA training corpus comprising 20 Newsgroups and SemCor data, with number of topics K = 40 and number of LDA bag-of-words L = 14.

6 Results

We now present the results on both English lexical sample task and all-words task.

6.1 Lexical Sample Task

With the optimal configurations from Senseval-2, we tested the systems on Senseval-3 data. Table 5 shows our F-measure result compared to some of the best reported systems. Although SVM with topic features shows limited success with only a 0.6% improvement, the Bayesian network approach has again demonstrated a good improvement of 3.8% over its baseline and is better than previous reported best systems except ASO(Ando, 2006).

Bayes (Soft Tag)	73.6
SVM-topic	73.0
SVM baseline	72.4
NB baseline	69.8
ASO(Ando, 2006)	74.1
SVM-LSA (Strapparava et al., 2004)	73.3
Senseval-3 Best System(Grozea, 2004)	72.9

Table 5: Results compared to previous best systems on Senseval-3 English lexical sample task.

6.2 All-words Task

The F-measure micro-averaged result for our systems as well as previous best systems for Senseval-2 and Senseval-3 all-words task are shown in Table 6 and Table 7 respectively. Bayesian network with soft tagging achieved 2.6% improvement over its baseline in Senseval-2 and 1.7% in Senseval-3. The results also rival some previous best systems, except for SMUaw (Mihalcea, 2002) which used additional labeled data.

Bayes (Soft Tag)	66.3
NB baseline	63.7
SMUaw (Mihalcea, 2002)	69.0
Simil-Prime (Kohomban and Lee, 2005)	66.4
Senseval-2 Best System (CNTS-Antwerp (Hoste et al., 2001))	63.6

Table 6: Results compared to previous best systems on Senseval-2 English all-words task.

Bayes (Soft Tag)	66.1
NB baseline	64.6
Simil-Prime (Kohomban and Lee, 2005)	66.1
Senseval-3 Best System (GAMBL-AW-S(Decadt et al., 2004))	65.2
Senseval-3 2nd Best System (SenseLearner (Mihalcea and Faruque, 2004))	64.6

Table 7: Results compared to previous best systems on Senseval-3 English all-words task.

6.3 Significance of Results

We perform the χ^2-test, using the Bayesian network and its naïve Bayes baseline (NB baseline) as pairs, to verify the significance of these results. The result is reported in Table 8. The results are significant at 90% confidence level, except for the Senseval-3 all-words task.

	Senseval-2	Senseval-3
All-word	0.0527	0.2925
Lexical Sample	$<$0.0001	0.0002

Table 8: P value for χ^2-test significance levels of results.

6.4 SVM with Topic Features

The results on lexical sample task show that SVM benefits less from the topic feature than the Bayesian approach. One possible reason is that SVM baseline is able to use all bag-of-words from surrounding context while naïve Bayes baseline can only use very few without decreasing its accuracy, due to the sparse representation. In this sense, SVM baseline already captures some of the topical information, leaving a smaller room for improvement. In fact, if we exclude the bag-of-words feature from the SVM baseline and add in the topic features, we are able to achieve almost the same accuracy as we did with both features included, as shown in Table 9. This further shows that the topic feature is a better representation of global context than the bag-of-words feature.

SVM baseline	72.4
SVM baseline - BAG + topic	73.5
SVM-topic	73.6

Table 9: Results on Senseval-3 English lexical sample task

6.5 Results on Different Parts-of-Speech

We analyse the result obtained on Senseval-3 English lexical sample task (using Senseval-2 optimal configuration) according to the test instance's part-of-speech, which includes noun, verb and adjective, compared to the naïve Bayes baseline. Table 10 shows the relative improvement on each part-of-speech. The second column shows the number of testing instances belonging to the particular part-of-speech. The third and fourth column shows the

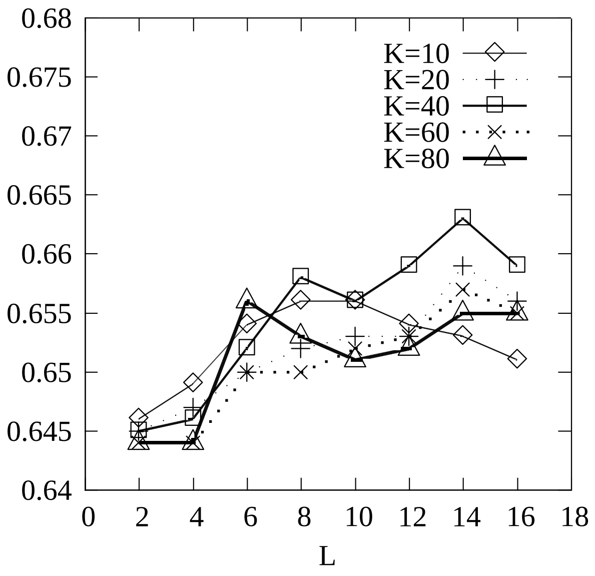


Figure 4: Accuracy with varing L and K on Senseval-2 all-words task

accuracy achieved by naïve Bayes baseline and the Bayesian network. Adjectives show no improvement while verbs show a moderate $+2.2\%$ improvement. Nouns clearly benefit from topical information much more than the other two parts-of-speech, obtaining a $+5.7\%$ increase over its baseline.

POS	Total	NB baseline	Bayes (Soft Tag)
Noun	1807	69.5	75.2
Verb	1978	71.1	73.5
Adj	159	57.2	57.2
Total	3944	69.8	73.6

Table 10: Improvement with different POS on Senseval-3 lexical sample task

6.6 Sensitivity to L and K

We tested on Senseval-2 all-words task using different L and K. Figure 4 is the result.

6.7 Results on SemEval-1

We participated in SemEval-1 English coarse-grained all-words task (task 7), English fine-grained all-words task (task 17, subtask 3) and English coarse-grained lexical sample task (task 17, subtask 1), using the method described in this paper. For all-words task, we use Senseval-2 and Senseval-3 all-words task data as our validation set to fine tune the parameters. For lexical sample task, we use the training data provided as the validation set.

We achieved 88.7%, 81.6% and 57.6% for coarse-grained lexical sample task, coarse-grained all-words task and fine-grained all-words task respectively. The results ranked first, second and fourth in the three tasks respectively.

7 Conclusion and Future Work

In this paper, we showed that by using LDA algorithm on bag-of-words feature, one can utilise more topical information and boost the classifiers accuracy on both English lexical sample and all-words task. Only unlabeled data is needed for this improvement. It would be interesting to see how the feature can help on WSD of other languages and other natural language processing tasks such as named-entity recognition.

References

Y. K. Lee and H. T. Ng. 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In *Proc. of EMNLP*.

B. Snyder and M. Palmer. 2004. The English All-Words Task. In *Proc. of Senseval-3*.

U. S. Kohomban and W. S. Lee 2005. Learning Semantic Classes for Word Sense Disambiguation. In *Proc. of ACL*.

R. K. Ando. 2006. Applying Alternating Structure Optimization to Word Sense Disambiguation. In *Proc. of CoNLL*.

Y. S. Chan and H. T. Ng 2005. Scaling Up Word Sense Disambiguation via Parallel Texts. In *Proc. of AAAI*.

R. K. Ando and T. Zhang. 2005a. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*.

R. K. Ando and T. Zhang. 2005b. A High-Performance Semi-Supervised Learning Method for Text Chunking. In *Proc. of ACL*.

P. Resnik and D. Yarowsky. 1997. A Perspective on Word Sense Disambiguation Methods and Their Evaluation. In *Proc. of ACL*.

D. M. Blei and A. Y. Ng and M. I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*.

A. Ratnaparkhi 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proc. of EMNLP*.

E. Charniak 2000. A Maximum-Entropy-Inspired Parser. In *Proc. of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

V. N. Vapnik 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York.

R. Florian and D. Yarowsky 2002. Modeling consensus: Classifier Combination for Word Sense Disambiguation. In *Proc. of EMNLP*.

D. Wu and W. Su and M. Carpuat. 2004. A Kernel PCA Method for Superior Word Sense Disambiguation. In *Proc. of ACL*.

C. Strapparava and A. Gliozzo and C. Giuliano 2004. Pattern Abstraction and Term Similarity for Word Sense Disambiguation: IRST at Senseval-3. In *Proc. of Senseval-3*.

C. Grozea 2004. Finding Optimal Parameter Settings for High Performance Word Sense Disambiguation. In *Proc. of Senseval-3*.

R. Mihalcea 2002. Bootstrapping Large Sense Tagged Corpora. In *Proc. of the 3rd International Conference on Languages Resources and Evaluations*.

V. Hoste and A. Kool and W. Daelmans 2001. Classifier Optimization and Combination in English All Words Task. In *Proc. of Senseval-2*.

B. Decadt and V. Hoste and W. Daelmans 2004. GAMBL, Genetic Algorithm Optimization of Memory-Based WSD. In *Proc. of Senseval-3*.

R. Mihalcea and E. Faruque 2004. Sense-learner: Minimally Supervised Word Sense Disambiguation for All Words in Open Text. In *Proc. of Senseval-3*.

A Topic Model for Word Sense Disambiguation

Jordan Boyd-Graber
Computer Science
Princeton University
Princeton, NJ 08540
jbg@princeton.edu

David Blei
Computer Science
Princeton University
Princeton, NJ 08540
blei@cs.princeton.edu

Xiaojin Zhu
Computer Science
University of Wisconsin
Madison, WI 53706
jerryzhu@cs.wisc.edu

Abstract

We develop latent Dirichlet allocation with WORDNET (LDAWN), an unsupervised probabilistic topic model that includes word sense as a hidden variable. We develop a probabilistic posterior inference algorithm for simultaneously disambiguating a corpus and learning the domains in which to consider each word. Using the WORDNET hierarchy, we embed the construction of Abney and Light (1999) in the topic model and show that automatically learned domains improve WSD accuracy compared to alternative contexts.

1 Introduction

Word sense disambiguation (WSD) is the task of determining the meaning of an ambiguous word in its context. It is an important problem in natural language processing (NLP) because effective WSD can improve systems for tasks such as information retrieval, machine translation, and summarization. In this paper, we develop latent Dirichlet allocation with WORDNET (LDAWN), a generative probabilistic topic model for WSD where the sense of the word is a hidden random variable that is inferred from data.

There are two central advantages to this approach. First, with LDAWN we automatically learn the context in which a word is disambiguated. Rather than disambiguating at the sentence-level or the document-level, our model uses the other words that share the same hidden topic across many documents.

Second, LDAWN is a fully-fledged generative model. Generative models are modular and can be easily combined and composed to form more complicated models. (As a canonical example, the ubiquitous hidden Markov model is a series of mixture models chained together.) Thus, developing a generative model for WSD gives other generative NLP algorithms a natural way to take advantage of the hidden senses of words.

In general, topic models are statistical models of text that posit a hidden space of topics in which the corpus is embedded (Blei et al., 2003). Given a corpus, posterior inference in topic models amounts to automatically discovering the underlying themes that permeate the collection. Topic models have recently been applied to information retrieval (Wei and Croft, 2006), text classification (Blei et al., 2003), and dialogue segmentation (Purver et al., 2006).

While topic models capture the polysemous use of words, they do not carry the explicit notion of *sense* that is necessary for WSD. LDAWN extends the topic modeling framework to include a hidden meaning in the word generation process. In this case, posterior inference discovers both the topics of the corpus and the meanings assigned to each of its words.

After introducing a disambiguation scheme based on probabilistic walks over the WORDNET hierarchy (Section 2), we embed the WORDNET-WALK in a topic model, where each topic is associated with walks that prefer different neighborhoods of WORDNET (Section 2.1). Then, we describe a Gibbs sampling algorithm for approximate posterior inference that learns the senses and topics that best explain a corpus (Section 3). Finally, we evaluate our system on real-world WSD data, discuss the properties of the topics and disambiguation accuracy results, and draw connections to other WSD algorithms from the research literature.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1024–1033, Prague, June 2007. ©2007 Association for Computational Linguistics

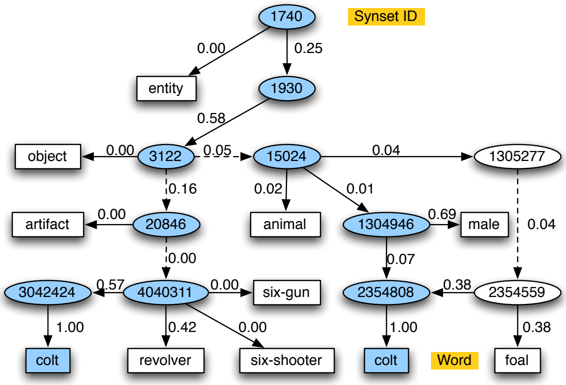


Figure 1: The possible paths to reach the word "colt" in WORDNET. Dashed lines represent omitted links. All words in the synset containing "revolver" are shown, but only one word from other synsets is shown. Edge labels are probabilities of transitioning from synset i to synset j. Note how this favors frequent terms, such as "revolver," over ones like "six-shooter."

2 Topic models and WordNet

The WORDNET-WALK is a probabilistic process of word generation that is based on the hyponomy relationship in WORDNET (Miller, 1990). WORDNET, a lexical resource designed by psychologists and lexicographers to mimic the semantic organization in the human mind, links "synsets" (short for synonym sets) with myriad connections. The specific relation we're interested in, hyponomy, points from general concepts to more specific ones and is sometimes called the "is-a" relationship.

As first described by Abney and Light (1999), we imagine an agent who starts at synset `[entity]`, which points to every noun in WORDNET 2.1 by some sequence of hyponomy relations, and then chooses the next node in its random walk from the hyponyms of its current position. The agent repeats this process until it reaches a leaf node, which corresponds to a single word (each of the synset's words are unique leaves of a synset in our construction). For an example of all the paths that might generate the word "colt" see Figure 1. The WORDNET-WALK is parameterized by a set of distributions over children for each synset s in WORDNET, $\boldsymbol{\beta}_s$.

Symbol	Meaning
K	number of topics
$\beta_{k,s}$	multinomial probability vector over the successors of synset s in topic k
S	scalar that, when multiplied by α_s gives the prior for $\beta_{k,s}$
α_s	normalized vector whose i^{th} entry, when multiplied by S, gives the prior probability for going from s to i
θ_d	multinomial probability vector over the topics that generate document d
τ	prior for θ
z	assignment of a word to a topic
Λ	a path assignment through WORDNET ending at a word.
$\lambda_{i,j}$	one link in a path λ going from synset i to synset j.

Table 1: A summary of the notation used in the paper. Bold vectors correspond to collections of variables (i.e. z_u refers to a topic of a single word, but $\boldsymbol{z}_{1:D}$ are the topics assignments of words in document 1 through D).

2.1 A topic model for WSD

The WORDNET-WALK has two important properties. First, it describes a random process for word generation. Thus, it is a distribution over words and thus can be integrated into any generative model of text, such as topic models. Second, the synset that produces each word is a hidden random variable. Given a word assumed to be generated by a WORDNET-WALK, we can use posterior inference to predict which synset produced the word.

These properties allow us to develop LDAWN, which is a fusion of these WORDNET-WALKs and latent Dirichlet allocation (LDA) (Blei et al., 2003), a probabilistic model of documents that is an improvement to pLSI (Hofmann, 1999). LDA assumes that there are K "topics," multinomial distributions over words, which describe a collection. Each document exhibits multiple topics, and each word in each document is associated with one of them.

Although the term "topic" evokes a collection of ideas that share a common theme and although the topics derived by LDA seem to possess semantic coherence, there is no reason to believe this would be true of the most likely multinomial distributions that could have created the corpus given the assumed generative model. That semantically similar words are likely to occur together is a byproduct of how language is actually used.

In LDAWN, we replace the multinomial topic distributions with a WORDNET-WALK, as described above. LDAWN assumes a corpus is generated by the following process (for an overview of the notation used in this paper, see Table 1).

1. For each topic, $k \in \{1, \dots, K\}$
   (a) For each synset s, randomly choose transition probabilities $\beta_{k,s} \sim \text{Dir}(S\boldsymbol{\alpha}_s)$.
2. For each document $d \in \{1, \dots, D\}$
   (a) Select a topic distribution $\theta_d \sim \text{Dir}(\tau)$
   (b) For each word $n \in \{1, \dots, N_d\}$
      i. Select a topic $z \sim \text{Mult}(1, \theta_d)$
      ii. Create a path $\Lambda_{d,n}$ starting with λ_0 as the root node.
      iii. From children of λ_i:
         A. Choose the next node in the walk $\lambda_{i+1} \sim \text{Mult}(1, \beta_{z,\lambda_i})$
         B. If λ_{i+1} is a leaf node, generate the associated word. Otherwise, repeat.

Every element of this process, including the synsets, is hidden except for the words of the documents. Thus, given a collection of documents, our goal is to perform *posterior inference*, which is the task of determining the conditional distribution of the hidden variables given the observations. In the case of LDAWN, the hidden variables are the parameters of the K WORDNET-WALKS, the topic assignments of each word in the collection, and the synset path of each word. In a sense, posterior inference reverses the process described above.

Specifically, given a document collection $\boldsymbol{w}_{1:D}$, the full posterior is

$$p(\boldsymbol{\beta}_{1:K}, \boldsymbol{z}_{1:D}, \boldsymbol{\theta}_{1:D}, \boldsymbol{\Lambda}_{1:D} \,|\, \boldsymbol{w}_{1:D}, \tau, S\boldsymbol{\alpha}) \propto \Big(\prod_{k=1}^{K} p(\boldsymbol{\beta}_k \,|\, S\boldsymbol{\alpha}) \prod_{d=1}^{D} p(\theta_d \,|\, \tau) \prod_{n=1}^{N_d} p(\Lambda_{d,n} \,|\, \boldsymbol{\beta}_{1:K}) p(w_{d,n} \,|\, \Lambda_{d,n}) \Big), \qquad (1)$$

where the constant of proportionality is the marginal likelihood of the observed data.

Note that by encoding the synset paths as a hidden variable, we have posed the WSD problem as a question of posterior probabilistic inference. Further note that we have developed an unsupervised model. No labeled data is needed to disambiguate a corpus. Learning the posterior distribution amounts to simultaneously decomposing a corpus into topics and its words into their synsets.

The intuition behind LDAWN is that the words in a topic will have similar meanings and thus share paths within WORDNET. For example, WORDNET has two senses for the word "colt;" one referring to a young male horse and the other to a type of handgun (see Figure 1).

Although we have no *a priori* way of knowing which of the two paths to favor for a document, we assume that similar concepts will also appear in the document. Documents with unambiguous nouns such as "six-shooter" and "smoothbore" would make paths that pass through the synset `[firearm, piece, small-arm]` more likely than those going through `[animal, animate being, beast, brute, creature, fauna]`. In practice, we hope to see a WORDNET-WALK that looks like Figure 2, which points to the right sense of cancer for a medical context.

LDAWN is a Bayesian framework, as each variable has a prior distribution. In particular, the Dirichlet prior for β_s, specified by a scaling factor S and a normalized vector α_s fulfills two functions. First, as the overall strength of S increases, we place a greater emphasis on the prior. This is equivalent to the need for balancing as noted by Abney and Light (1999).

The other function that the Dirichlet prior serves is to enable us to encode any information we have about how we suspect the transitions to children nodes will be distributed. For instance, we might expect that the words associated with a synset will be produced in a way roughly similar to the token probability in a corpus. For example, even though "meal" might refer to both ground cereals or food eaten at a single sitting and "repast" exclusively to the latter, the synset `[meal, repast, food eaten at a single sitting]` still prefers to transition to "meal" over "repast" given the overall corpus counts (see Figure 1, which shows prior transition probabilities for "revolver").

By setting $\alpha_{s,i}$, the prior probability of transitioning from synset s to node i, proportional to the total number of observed tokens in the children of i, we introduce a probabilistic variation on information content (Resnik, 1995). As in Resnik's definition, this value for non-word nodes is equal to the sum of all the frequencies of hyponym words. Unlike Resnik, we do not divide frequency among all senses of a word; each sense of a word contributes its full frequency to α.

3 Posterior Inference with Gibbs Sampling

As described above, the problem of WSD corresponds to posterior inference: determining the probability distribution of the hidden variables given observed words and then selecting the synsets of the most likely paths as the correct sense. Directly computing this posterior distribution, however, is not tractable because of the difficulty of calculating the normalizing constant in Equation 1.

To approximate the posterior, we use Gibbs sampling, which has proven to be a successful approximate inference technique for LDA (Griffiths and Steyvers, 2004). In Gibbs sampling, like all Markov chain Monte Carlo methods, we repeatedly sample from a Markov chain whose stationary distribution is the posterior of interest (Robert and Casella, 2004). Even though we don't know the full posterior, the samples can be used to form an empirical estimate of the target distribution. In LDAWN, the samples contain a configuration of the latent semantic states of the system, revealing the hidden topics and paths that likely led to the observed data.

Gibbs sampling reproduces the posterior distribution by repeatedly sampling each hidden variable conditioned on the current state of the other hidden variables and observations. More precisely, the state is given by a set of assignments where each word is assigned to a path through one of K WORDNET-WALK topics: u^{th} word w_u has a topic assignment z_u and a path assignment Λ_u. We use $\boldsymbol{z}_{-u}$ and $\boldsymbol{\Lambda}_{-u}$ to represent the topic and path assignments of all words except for u, respectively.

Sampling a new topic for the word w_u requires us to consider all of the paths that w_u can take in each topic and the topics of the other words in the document u is in. The probability of w_u taking on topic i is proportional to

$$p(z_u = i \,|\, \boldsymbol{z}_{-u}) \textstyle\sum_{\lambda} p(\lambda \,|\, \boldsymbol{\Lambda}_{-u}) \mathbb{1}[w_u \in \lambda], \qquad (2)$$

which is the probability of selecting z from θ_d times the probability of a path generating w_u from a path in the i^{th} WORDNET-WALK.

The first term, the topic probability of the u^{th} word, is based on the assignments to the K topics for words other than u in this document,

$$p(z_u = i | \boldsymbol{z}_{-u}) = \frac{n^{(d)}_{-u,i} + \tau_i}{\sum_j n^{(d)}_{-u,j} + \sum_{j=1}^{K} \tau_j}, \qquad (3)$$

where $n^{(d)}_{-u,j}$ is the number of words other than u in topic j for the document d that u appears in.

The second term in Equation 2 is a sum over the probabilities of every path that could have generated the word w_u. In practice, this sum can be computed using a dynamic program for all nodes that have unique parent (i.e. those that can't be reached by more than one path). Although the probability of a path is specific to the topic, as the transition probabilities for a synset are different across topics, we will omit the topic index in the equation,

$$p(\Lambda_u = \lambda | \boldsymbol{\Lambda}_{-u},) = \textstyle\prod_{i=1}^{l-1} \beta^{-u}_{\lambda_i,\lambda_{i+1}}. \qquad (4)$$

3.1 Transition Probabilities

Computing the probability of a path requires us to take a product over our estimate of the probability from transitioning from i to j for all nodes i and j in the path λ. The other path assignments within this topic, however, play an important role in shaping the transition probabilities.

From the perspective of a single node i, only paths that pass through that node affect the probability of u also passing through that node. It's convenient to have an explicit count of all of the paths that transition from i to j in this topic's WORDNET-WALK, so we use $T^{-u}_{i,j}$ to represent all of the paths that go from i to j in a topic other than the path currently assigned to u.

Given the assignment of all other words to paths, calculating the probability of transitioning from i to j with word u requires us to consider the prior α and the observations $T_{i,j}$ in our estimate of the expected value of the probability of transitioning from i to j,

$$\beta^{-u}_{i,j} = \frac{T^{-u}_{i,j} + S_i \alpha_{i,j}}{S_i + \sum_k T^{-u}_{i,k}}. \qquad (5)$$

As mentioned in Section 2.1, we paramaterize the prior for synset i as a vector α_i, which sums to one, and a scale parameter S.

The next step, once we've selected a topic, is to select a path within that topic. This requires the computation of the path probabilities as specified in Equation 4 for all of the paths w_u can take in the sampled topic and then sampling from the path probabilities.

The Gibbs sampler is essentially a randomized hill climbing algorithm on the posterior likelihood as a function of the configuration of hidden variables. The numerator of Equation 1 is proportional to that posterior and thus allows us to track the sampler's progress. We assess convergence to a local mode of the posterior by monitoring this quantity.

4 Experiments

In this section, we describe the properties of the topics induced by running the previously described Gibbs sampling method on corpora and how these topics improve WSD accuracy.

Of the two data sets used during the course of our evaluation, the primary dataset was SEMCOR (Miller et al., 1993), which is a subset of the Brown corpus with many nouns manually labeled with the correct WORDNET sense. The words in this dataset are lemmatized, and multi-word expressions that are present in WORDNET are identified. Only the words in SEMCOR were used in the Gibbs sampling procedure; the synset assignments were only used for assessing the accuracy of the final predictions.

We also used the British National Corpus, which is not lemmatized and which does not have multi-word expressions. The text was first run through a lemmatizer, and then sequences of words which matched a multi-word expression in WORDNET were joined together into a single word. We took nouns that appeared in SEMCOR twice or in the BNC at least 25 times and used the BNC to compute the information-content analog $\boldsymbol{\alpha}$ for individual nouns (For example, the probabilities in Figure 1 correspond to $\boldsymbol{\alpha}$).

4.1 Topics

Like the topics created by structures such as LDA, the topics in Table 2 coalesce around reasonable themes. The word list was compiled by summing over all of the possible leaves that could have generated each of the words and sorting the words by decreasing probability. In the vast majority of cases, a single synset's high probability is responsible for the words' positions on the list.

Reassuringly, many of the top senses for the present words correspond to the most frequent sense in SEMCOR. For example, in Topic 4, the senses for "space" and "function" correspond to the top senses in SEMCOR, and while the top sense for "set" corresponds to "an abstract collection of numbers or symbols" rather than "a group of the same kind that belong together and are so used," it makes sense given the math-based words in the topic. "Point," however, corresponds to the sense used in the phrase "I got to the point of boiling the water," which is neither the top SEMCOR sense nor a sense which makes sense given the other words in the topic.

While the topics presented in Table 2 resemble the topics one would obtain through models like LDA (Blei et al., 2003), they are not identical. Because of the lengthy process of Gibbs sampling, we initially thought that using LDA assignments as an initial state would converge faster than a random initial assignment. While this was the case, it converged to a state that less probable than the randomly initialized state and no better at sense disambiguation (and sometimes worse). The topics presented in 2 represent words both that co-occur together in a corpus and co-occur on paths through WORDNET. Because topics created through LDA only have the first property, they usually do worse in terms of both total probability and disambiguation accuracy (see Figure 3).

Another interesting property of topics in LDAWN is that, with higher levels of smoothing, words that don't appear in a corpus (or appear rarely) but are in similar parts of WORDNET might have relatively high probability in a topic. For example, "maturity" in topic two in Table 2 is sandwiched between "foot" and "center," both of which occur about five times more than "maturity." This might improve LDA-based information retrieval schemes (Wei and Croft, 2006) .

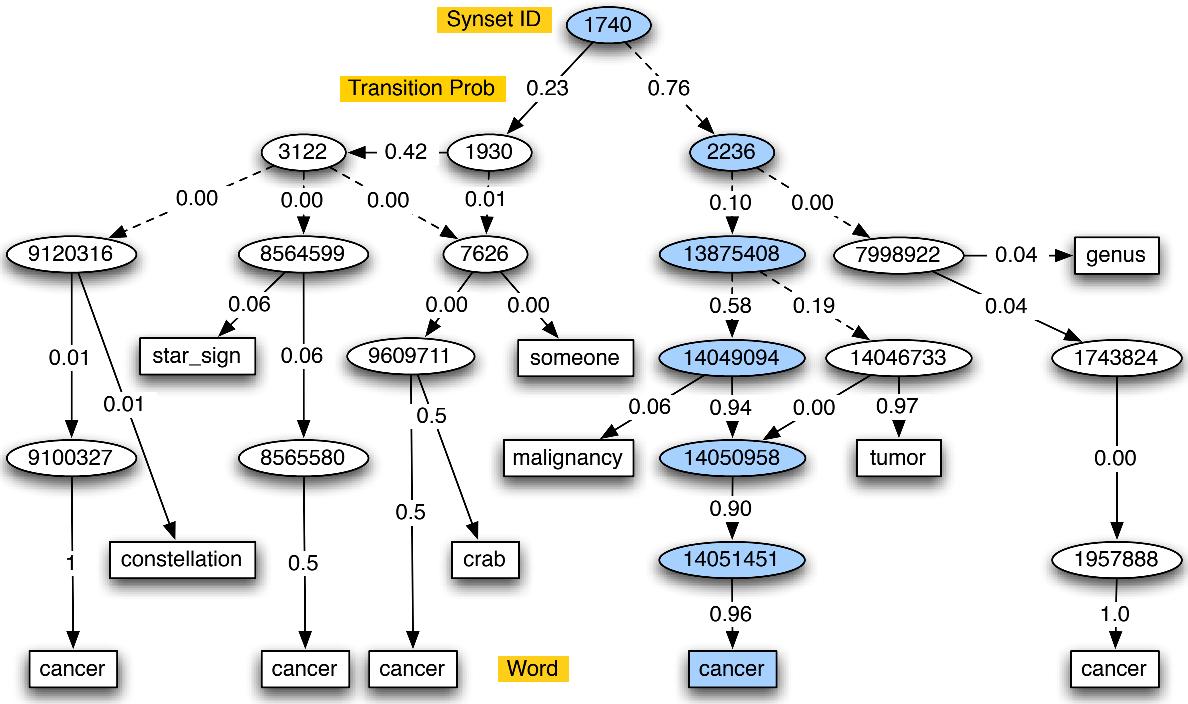


Figure 2: The possible paths to reach the word "cancer" in WORDNET along with transition probabilities from the medically-themed Topic 2 in Table 2, with the most probable path highlighted. The dashed lines represent multiple links that have been consolidated, and synsets are represented by their offsets within WORDNET 2.1. Some words for immediate hypernyms have also been included to give context. In all other topics, the person, animal, or constellation senses were preferred.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
president	growth	material	point	water	plant	music
party	age	object	number	house	change	film
city	treatment	color	value	road	month	work
election	feed	form	function	area	worker	life
administration	day	subject	set	city	report	time
official	period	part	square	land	mercer	world
office	head	self	space	home	requirement	group
bill	portion	picture	polynomial	farm	bank	audience
yesterday	length	artist	operator	spring	farmer	play
court	level	art	component	bridge	production	thing
meet	foot	patient	corner	pool	medium	style
police	maturity	communication	direction	site	petitioner	year
service	center	movement	curve	interest	relationship	show

Table 2: The most probable words from six randomly chosen WORDNET-walks from a thirty-two topic model trained on the words in SEMCOR. These are summed over all of the possible synsets that generate the words. However, the vast majority of the contributions come from a single synset.

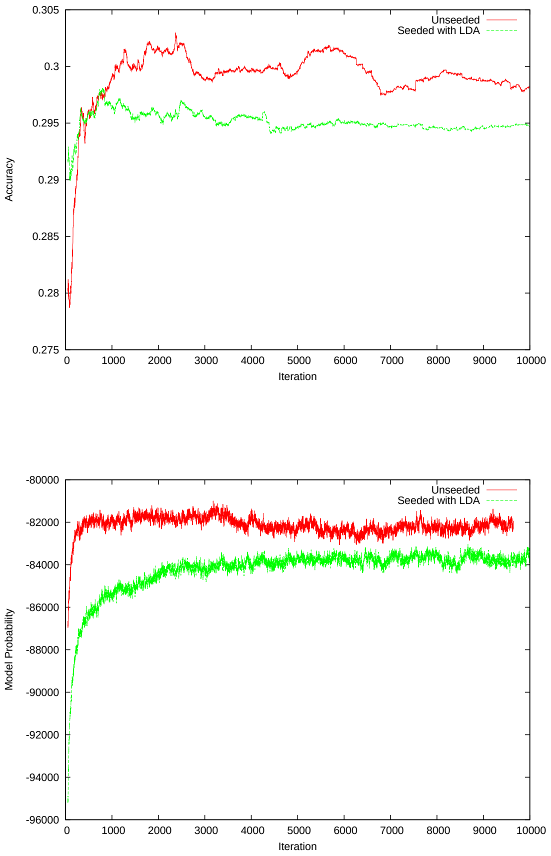


Figure 3: Topics seeded with LDA initially have a higher disambiguation accuracy, but are quickly matched by unseeded topics. The probability for the seeded topics starts lower and remains lower.

4.2 Topics and the Weight of the Prior

Because the Dirichlet smoothing factor in part determines the topics, it also affects the disambiguation. Figure 4 shows the modal disambiguation achieved for each of the settings of $S = \{0.1, 1, 5, 10, 15, 20\}$. Each line is one setting of K and each point on the line is a setting of S. Each data point is a run for the Gibbs sampler for 10,000 iterations. The disambiguation, taken at the mode, improved with moderate settings of S, which suggests that the data are still sparse for many of the walks, although the improvement vanishes if S dominates with much larger values. This makes sense, as each walk has over 100,000 parameters, there are fewer than 100,000 words in SEMCOR, and each word only serves as evidence to at most 19 parameters (the length of the longest path in WORDNET).

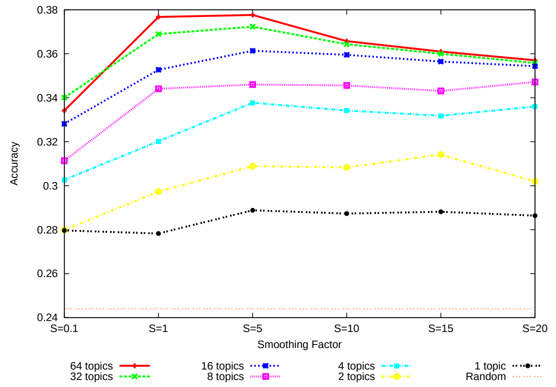


Figure 4: Each line represents experiments with a set number of topics and variable amounts of smoothing on the SEMCOR corpus. The random baseline is at the bottom of the graph, and adding topics improves accuracy. As smoothing increases, the prior (based on token frequency) becomes stronger. Accuracy is the percentage of correctly disambiguated polysemous words in SEMCOR at the mode.

Generally, a greater number of topics increased the accuracy of the mode, but after around sixteen topics, gains became much smaller. The effect of α is also related to the number of topics, as a value of S for a very large number of topics might overwhelm the observed data, while the same value of S might be the perfect balance for a smaller number of topics. For comparison, the method of using a WORDNET-WALK applied to smaller contexts such as sentences or documents achieves an accuracy of between 26% and 30%, depending on the level of smoothing.

5 Error Analysis

This method works well in cases where the delineation can be readily determined from the overall topic of the document. Words such as "kid," "may," "shear," "coach," "incident," "fence," "bee," and (previously used as an example) "colt" were all perfectly disambiguated by this method. Figure 2 shows the WORDNET-WALK corresponding to a medical topic that correctly disambiguates "cancer."

Problems arose, however, with highly frequent words, such as "man" and "time" that have many senses and can occur in many types of documents. For example, "man" can be associated with many possible meanings: island, game equipment, servant, husband, a specific mammal, etc.

Although we know that the "adult male" sense should be preferred, the alternative meanings will also be likely if they can be assigned to a topic that shares common paths in WORDNET; the documents contain, however, many other places, jobs, and animals which are reasonable explanations (to LDAWN) of how "man" was generated. Unfortunately, "man" is such a ubiquitous term that topics, which are derived from the frequency of words within an entire document, are ultimately uninformative about its usage.

While mistakes on these highly frequent terms significantly hurt our accuracy, errors associated with less frequent terms reveal that WORDNET's structure is not easily transformed into a probabilistic graph. For instance, there are two senses of the word "quarterback," a player in American football. One is position itself and the other is a person playing that position. While one would expect co-occurrence in sentences such as "quarterback is a easy position, so our quarterback is happy," the paths to both terms share only the root node, thus making it highly unlikely a topic would cover both senses.

Because of WORDNET's breadth, rare senses also impact disambiguation. For example, the metonymical use of "door" to represent a whole building as in the phrase "girl next door" is under the same parent as sixty other synsets containing "bridge," "balcony," "body," "arch," "floor," and "corner." Surrounded by such common terms that are also likely to co-occur with the more conventional meanings of door, this very rare sense becomes the preferred disambiguation of "door."

6 Related Work

Abney and Light's initial probabilistic WSD approach (1999) was further developed into a Bayesian network model by Ciaramita and Johnson (2000), who likewise used the appearance of monosemous terms close to ambiguous ones to "explain away" the usage of ambiguous terms in selectional restrictions. We have adapted these approaches and put them into the context of a topic model.

Recently, other approaches have created *ad hoc* connections between synsets in WORDNET and then considered walks through the newly created graph. Given the difficulties of using existing connections in WORDNET, Mihalcea (2005) proposed creating links between adjacent synsets that might comprise a sentence, initially setting weights to be equal to the Lesk overlap between the pairs, and then using the PageRank algorithm to determine the stationary distribution over synsets.

6.1 Topics and Domains

Yarowsky was one of the first to contend that "there is one sense for discourse" (1992). This has lead to the approaches like that of Magnini (Magnini et al., 2001) that attempt to find the category of a text, select the most appropriate synset, and then assign the selected sense using domain annotation attached to WORDNET.

LDAWN is different in that the categories are not an *a priori* concept that must be painstakingly annotated within WORDNET and require no augmentation of WORDNET. This technique could indeed be used with any hierarchy. Our concepts are the ones that best partition the space of documents and do the best job of describing the distinctions of diction that separate documents from different domains.

6.2 Similarity Measures

Our approach gives a probabilistic method of using information content (Resnik, 1995) as a starting point that can be adjusted to cluster words in a given topic together; this is similar to the Jiang-Conrath similarity measure (1997), which has been used in many applications in addition to disambiguation. Patwardhan (2003) offers a broad evaluation of similarity measures for WSD.

Our technique for combining the cues of topics and distance in WORDNET is adjusted in a way similar in spirit to Buitelaar and Sacaleanu (2001), but we consider the appearance of a single term to be evidence for not just that sense and its immediate neighbors in the hyponomy tree but for all of the sense's children and ancestors.

Like McCarthy (2004), our unsupervised system acquires a single predominant sense for a domain based on a synthesis of information derived from a textual corpus, topics, and WORDNET-derived similarity, a probabilistic information content measure. By adding syntactic information from a thesaurus derived from syntactic features (taken from Lin's automatically generated thesaurus (1998)), McCarthy achieved 48% accuracy in a similar evaluation on SEMCOR; LDAWN is thus substantially less effective in disambiguation compared to state-of-the-art methods. This suggests, however, that other methods might be improved by adding topics and that our method might be improved by using more information than word counts.

7 Conclusion and Future Work

The LDAWN model presented here makes two contributions to research in automatic word sense disambiguation. First, we demonstrate a method for automatically partitioning a document into topics that includes explicit semantic information. Second, we show that, at least for one simple model of WSD, embedding a document in probabilistic latent structure, i.e., a "topic," can improve WSD.

There are two avenues of research with LDAWN that we will explore. First, the statistical nature of this approach allows LDAWN to be used as a component in larger models for other language tasks. Other probabilistic models of language could insert the ability to query synsets or paths of WORDNET. Similarly, any topic based information retrieval scheme could employ topics that include semantically relevant (but perhaps unobserved) terms. Incorporating this model in a larger syntactically-aware model, which could benefit from the local context as well as the document level context, is an important component of future research.

Second, the results presented here show a marked improvement in accuracy as more topics are added to the baseline model, although the final result is not comparable to state-of-the-art techniques. As most errors were attributable to the hyponomy structure of WORDNET, incorporating the novel use of topic modeling presented here with a more mature unsupervised WSD algorithm to replace the underlying WORDNET-WALK could lead to advances in state-of-the-art unsupervised WSD accuracy.

References

Steven Abney and Marc Light. 1999. Hiding a semantic hierarchy in a markov model. In *Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing*, pages 1–8.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Paul Buitelaar and Bogdan Sacaleanu. 2001. Ranking and selecting synsets by domain relevance. In *Proceedings of WordNet and Other Lexical Resources: Applications, Extensions and Customizations. NAACL 2001.* Association for Computational Linguistics.

Massimiliano Ciaramita and Mark Johnson. 2000. Explaining away ambiguity: Learning verb selectional preference with bayesian networks. In *COLING-00*, pages 187–193.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 233–237. Association for Computational Linguistics.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, pages 5228–5235.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. *Proceedings of the Twenty-Second Annual International SIGIR Conference*.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings on International Conference on Research in Computational Linguistics*, Taiwan.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304.

Bernardo Magnini, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo. 2001. Using domain information for word sense disambiguation. In *In Proceedings of 2^{nd} International Workshop on Evaluating Word Sense Disambiguation Systems*, Toulouse, France.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *In 42nd Annual Meeting of the Association for Computational Linguistics*, pages 280–287.

Rada Mihalcea. 2005. Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the Joint Human Language Technology and Empirical Methods in Natural Language Processing Conference*, pages 411–418.

George Miller, Claudia Leacock, Randee Tengi, and Ross Bunker. 1993. A semantic concordance. In *3rd DARPA Workshop on Human Language Technology*, pages 303–308.

George A. Miller. 1990. Nouns in WordNet: A lexical inheritance system. *International Journal of Lexicography*, 3(4):245–264.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. Using Measures of Semantic Relatedness for Word Sense Disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257.

Matthew Purver, Konrad Körding, Thomas Griffiths, and Joshua Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of COLING-ACL*.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *International Joint Conferences on Artificial Intelligence*, pages 448–453.

Christian Robert and George Casella. 2004. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer-Verlag, New York, NY.

Xing Wei and Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the Twenty-Ninth Annual International SIGIR Conference*.

Validation and Evaluation of Automatically Acquired Multiword Expressions for Grammar Engineering

Aline Villavicencio♣♠, Valia Kordoni♢, Yi Zhang♢,
Marco Idiart♡ and Carlos Ramisch♣

♣Institute of Informatics, Federal University of Rio Grande do Sul (Brazil)
♠Department of Computer Sciences, Bath University (UK)
♢Department of Computational Linguistics, Saarland University, and DFKI GmbH (Germany)
♡Institute of Physics, Federal University of Rio Grande do Sul (Brazil)
avillavicencio@inf.ufrgs.br, {yzhang,kordoni}@coli.uni-sb.de
idiart@if.ufrgs.br, ceramisch@inf.ufrgs.br

Abstract

This paper focuses on the evaluation of methods for the automatic acquisition of Multiword Expressions (MWEs) for robust grammar engineering. First we investigate the hypothesis that MWEs can be detected by the distinct statistical properties of their component words, regardless of their type, comparing 3 statistical measures: mutual information (MI), χ^2 and permutation entropy (PE). Our overall conclusion is that at least two measures, MI and PE, seem to differentiate MWEs from non-MWEs. We then investigate the influence of the size and quality of different corpora, using the BNC and the Web search engines Google and Yahoo. We conclude that, in terms of language usage, web generated corpora are fairly similar to more carefully built corpora, like the BNC, indicating that the lack of control and balance of these corpora are probably compensated by their size. Finally, we show a qualitative evaluation of the results of automatically adding extracted MWEs to existing linguistic resources. We argue that such a process improves qualitatively, if a more compositional approach to grammar/lexicon automated extension is adopted.

1 Introduction

The task of automatically identifying Multiword Expressions (MWEs) like phrasal verbs (*break down*) and compound nouns (*coffee machine*) using statistical measures has been the focus of considerable investigative effort, (e.g. Pearce (2002), Evert and Krenn (2005) and Zhang et al. (2006)). Given the heterogeneousness of the different phenomena that are considered to be MWEs, there is no consensus about which method is best suited for which type of MWE, and if there is a single method that can be successfully used for any kind of MWE.

Another difficulty for work on MWE identification is that of the evaluation of the results obtained (Pearce, 2002; Evert and Krenn, 2005), starting from the lack of consensus about a precise definition for MWEs (Villavicencio et al., 2005).

In this paper we investigate some of the issues involved in the evaluation of automatically extracted MWEs, from their extraction to their subsequent use in an NLP task. In order to do that, we present a discussion of different statistical measures, and the influence that the size and quality of different data sources have. We then perform a comparison of these measures and discuss whether there is a single measure that has good overall performance for MWEs in general, regardless of their type. Finally, we perform a qualitative evaluation of the results of adding automatically extracted MWEs to a linguistic resource, taking as basis for the evaluation the approach proposed by Zhang et al. (2006). We argue that such results can improve in quality if a more compositional approach to MWE encoding is adopted for the grammar extension. Having more accurate means of deciding for an appropriate method for identifying and incorporating MWEs is critical for maintaining the quality of linguistic resources for precise NLP.

This paper starts with a discussion of MWEs (§ 2), of their coverage in linguistic resources (§ 3), and of some methods proposed for automatically identifying them (§ 4). This is followed by a detailed investigation and comparison of measures for MWE identification (§ 5).

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1034–1043, Prague, June 2007. ©2007 Association for Computational Linguistics

After that we present an approach for predicting appropriate lexico-syntactic categories for their inclusion in a linguistic resource, and an evaluation of the results in a parsing task(§ 7). We finish with some conclusions and discussion of future work.

2 Multiword Expressions

The term Multiword Expressions has been used to describe expressions for which the syntactic or semantic properties of the whole expression cannot be derived from its parts (Sag et al., 2002), including a large number of related but distinct phenomena, such as phrasal verbs (e.g. *come along*), nominal compounds (e.g. *frying pan*), institutionalised phrases (e.g. *bread and butter*), and many others. Jackendoff (1997) estimates the number of MWEs in a speaker's lexicon to be comparable to the number of single words. However, due to their heterogeneous characteristics, MWEs present a tough challenge for both linguistic and computational work (Sag et al., 2002). For instance, some MWEs are fixed, and do not present internal variation, such as *ad hoc*, while others allow different degrees of internal variability and modification, such as *spill beans* (*spill several/musical/mountains of beans*).

Sag et al. (2002) discuss two main approaches commonly employed in NLP for treating MWEs: the *words-with-spaces* approach models an MWE as a single lexical entry and it can adequately capture fixed MWEs like *by and large*. A *compositional* approach treats MWEs by general and compositional methods of linguistic analysis, being able to capture more syntactically flexible MWEs, like *rock boat*, which cannot be satisfactorily captured by a words-with-spaces approach, since it would require lexical entries to be added for all the possible variations of an MWE (e.g. *rock/rocks/rocking this/that/his... boat*). Therefore, to provide a unified account for the detection and encoding of these distinct but related phenomena is a real challenge for NLP systems.

3 Grammar and Lexicon Coverage in Deep Processing

Many NLP tasks and applications, like Parsing and Machine Translation, depend on large-scale linguistic resources, such as electronic dictionaries and grammars for precise results. Several substantial resources exist: e.g., hand-crafted large-scale grammars like the English Resource Grammar (ERG - Flickinger (2000)) and the Dutch Alpino Grammar (Bouma et al., 2001).

Unfortunately, the construction of these resources is the manual result of human efforts and therefore likely to contain errors of omission and commission (Briscoe and Carroll, 1997). Furthermore, due to the open-ended and dynamic nature of languages, such linguistic resources are likely to be incomplete, and manual encoding of new entries and constructions is labour-intensive and costly.

Take, for instance, the coverage test results for the ERG (a broad-coverage precision HPSG grammar for English) on the British National Corpus (BNC). Baldwin et al. (2004), among many others, have investigated the main causes of parse failure, parsing a random sample of 20,000 strings from the written component of the BNC using the ERG. They have found that the large majority of failures is caused by missing lexical entries, with 40% of the cases, and missing constructions, with 39%, where missing MWEs accounted for 8% of total errors. That is, even by a margin, the lexical coverage is lower than the grammar construction coverage.

This indicates the acute need for robust (semi-)automated ways of acquiring lexical information for MWEs, and this is the one of the goals of this work. In the next section we discuss some approaches that have been developed in recent years to (semi-)automatically detect and/or repair lexical and grammar errors in linguistic grammars and/or extend their coverage.

4 Acquiring MWEs

The automatic acquisition of specific types of MWE has attracted much interest (Pearce, 2002; Baldwin and Villavicencio, 2002; Evert and Krenn, 2005; Villavicencio, 2005; van der

Beek, 2005; Nicholson and Baldwin, 2006). For instance, Baldwin and Villavicencio (2002) proposed a combination of methods to extract Verb-Particle Constructions (VPCs) from unannotated corpora, that in an evaluation on the Wall Street Journal achieved 85.9% precision and 87.1% recall. Nicholson and Baldwin (2006) investigated the prediction of the inherent semantic relation of a given compound nominalization using as statistical measure the confidence interval.

On the other hand, Zhang et al. (2006) looked at MWEs in general investigating the semi-automated detection of MWE candidates in texts using error mining techniques and validating them using a combination of the World Wide Web as a corpus and some statistical measures. 6248 sentences were then extracted from the BNC; these contained at least one of the 311 MWE candidates verified with World Wide Web in the way described in Zhang et al. (2006). For each occurrence of the MWE candidates in this set of sentences, the lexical type predictor proposed in Zhang and Kordoni (2006) predicted a lexical entry candidate. This resulted in 373 additional MWE lexical entries for the ERG grammar using a words-with-spaces approach. As reported in Zhang et al. (2006), this addition to the grammar resulted in a significant increase in grammar coverage of 14.4%. However, no further evaluation was done of the results of the measures used on the identification of MWEs or of the resulting grammar, as not all MWEs can be correctly handled by the simple words-with-spaces approach (Sag et al., 2002). And these are the starting points of the work we are reporting on here.

5 Evaluation of the Identification of MWEs

One way of viewing the MWE identification task is, given a list of sequences of words, to distinguish those that are genuine MWEs (e.g. *in the red*), from those that are just sequences of words that do not form any kind of meaningful unit (e.g. *of alcohol and*). In order to do that, one commonly used approach is to employ statistical measures (e.g. Pearce (2002) for collocations and Zhang et al. (2006) for MWEs in general). When dealing with statistical analysis there are two important statistical questions that should be addressed: *How reliable is the corpus used?* and *How precise is the chosen statistical measure to distinguish the phenomena studied?*.

In this section we look at these issues, for the particular case of trigrams, by testing different corpora and different statistical measures. For that we use 1039 trigrams that are the output of Zhang et al. (2006) error mining system, and frequencies collected from the BNC and from the World Wide Web. The former were collected from two different portions of the BNC, namely the fragment of the BNC (BNC$_f$) used in the error-mining experiments, and the complete BNC (from the site http://pie.usna.edu/), to test whether a larger sample of a more homogeneous and well balanced corpus improves results significantly. For the latter we used two different search engines: Google and Yahoo, and the frequencies collected reflect the number of pages that had exact matches of the n-grams searched, using the API tools for each engine.

5.1 Comparing Corpora

A corpus for NLP related work should be a reliable sample of the linguistic output of a given language. For this work in particular, we expect that the relative ordering in frequency for different n-grams is preserved across corpora, in the same domain (e.g. a corpus of chemistry articles). For, if this is not the case, different conclusions are certain to be drawn from different corpora.

The first test we performed was a direct comparison of the rank plots of the relative frequency of trigrams for the four corpora. We ranked 1039 MWE-candidate trigrams according to their occurrence in each corpus and we normalised this value by the total number of times any one of the 1039 trigrams appeared for each corpus. These normalisation values were: 66,101 times in BNC$_f$, 322,325 in BNC, 224,479,065 in Google and 6,081,786,313 in Yahoo. It is possible to have an estimate of the size of each corpus from these numbers: the trigrams account for something like 0.3% of the BNC corpora, while for Google and Yahoo nothing can be said since their sizes are not reliable numbers. Figure 1 displays the results. The overall ranking distribution is very similar for these corpora showing the expected Zipf like behaviour in spite of their different sizes.

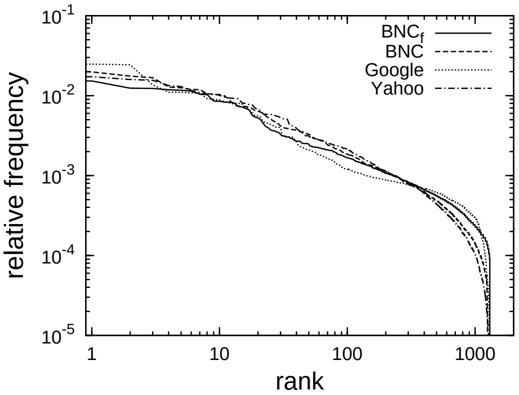


Figure 1: Relative frequency rank for the 1039 trigrams analysed.

Of course, the information coming from Figure 1 is not sufficient for our purposes. The order of the trigrams could be very different inside each corpus. Therefore a second test is needed to compare the rankings of the n-grams in each corpus. In order to do that we measure the Kendall's τ scores between corpora. Kendall's τ is a non-parametric method for estimating correlation between datasets (Press et al., 1992). For the number of trigrams studied here the Kendall's scores obtained imply a significant correlation between the corpora with p<0.000001. The significance indicates that the data are correlated and the *null* hypothesis of statistical independence is certainly disproved. Unfortunately disproving the *null* hypothesis does not give much information about the degree of correlation; it only asserts that it exists. Thus, it could be a very insignificant correlation. In table 1, we display a more intuitive measure to estimate the correlation, the probability Q that any 2 trigrams chosen from two corpora have the same relative ordering in frequency. This probability is related to Kendall's τ through the expression Q = $(1 + \tau)/2$.

	BNC	Google	Yahoo
BNC$_f$	0.81	0.73	0.78
BNC		0.73	0.77
Google			0.86

Table 1: The probability Q of 2 trigrams having the same frequency rank order for different corpora.

The results show that the four corpora are certainly correlated, and can probably be used interchangeably to access most of the statistical properties of the trigrams. Interestingly, a higher correlation was observed between Yahoo and Google than between BNC$_f$ and BNC, even though BNC$_f$ is a fragment of BNC, and therefore would be expected to have a very high correlation. This suggests that as corpora sizes increase, so do the correlations between them, meaning that they are more likely to agree on the ranking of a given MWE.

5.2 Comparing statistical measures - are they equivalent?

Here we concentrate on a single corpus, BNC$_f$, and compare the three statistical measures for MWE identification: Mutual Information (MI), χ^2 and Permutation Entropy (PE)(Zhang et al., 2006), to investigate if they order the trigrams in the same fashion.

MI and χ^2 are typical measures of association that compare the joint probability of occurrence of a certain group of events $p(abc)$ with a prediction derived from the *null* hypothesis of statistical independence between these events $p_\emptyset(abc) = p(a)p(b)p(c)$ (Press et al., 1992). In our case the events are the occurrences of words in a given position in an n-gram. For a trigram with words $w_1 w_2 w_3$, χ^2 is calculated as:

$$\chi^2 = \sum_{a,b,c} \frac{[\, n(abc) - n_\emptyset(abc)\,]^2}{n_\emptyset(abc)}$$

where a corresponds either to the word w_1 or to $\neg w_1$ (all but the word w_1) and so on. $n(abc)$ is the number of trigrams abc in the corpus, $n_\emptyset(abc) = n(a)n(b)n(c)/N^2$ is the predicted number from the *null* hypothesis, $n(a)$ is the number of unigrams a, and N the number of words in the corpus. Mutual Information, in terms of these numbers, is:

$$\mathrm{MI} \;=\; \sum_{a,b,c} \frac{n(abc)}{N} \, \log_2 \left[\frac{n(abc)}{n_{\emptyset}(abc)} \right]$$

The third measure, permutation entropy, is a measure of order association. Given the words w_1, w_2, and w_3, PE is calculated in this work as:

$$\mathrm{PE} = -\sum_{(i,j,k)} p(w_i w_j w_k) \ln \left[\, p(w_i w_j w_k) \,\right]$$

where the sum runs over all the permutations of the indexes and, therefore, over all possible positions of the selected words in the trigram. The probabilities are estimated from the number of occurrences of each permutation of a trigram (e.g. *by and large, large by and, and large by, and by large, large and by*, and *by large and*) as:

$$p(w_1 w_2 w_3) = \frac{n(w_1 w_2 w_3)}{\sum_{(i,j,k)} n(w_i w_j w_k)}$$

PE was proposed by Zhang et al. (2006) as a possible measure to detect MWEs, under the hypothesis that MWEs are more rigid to permutations and therefore present smaller PEs. Even though it is quite different from MI and χ^2, PE can also be thought as an indirect measure of statistical independence, since the more independent the words are the closer PE is from its maximal value ($\ln 6$, for trigrams). One possible advantage of this measure over the others is that it does not rely on single word counts, which are less accurate in Web based corpora.

Given the rankings produced for each one of these three measures we again use Kendall's τ test to assess correlation and its significance. Table 2 displays the Q probability of finding the same ordering in these three measures. The general conclusion from the table is that even though there is statistical significance in the correlations found (the p values are not displayed, but they are very low as before) the different measures order the trigrams very differently. There is a 70% chance of getting the same order from MI and χ^2, but it is safe to say that these measures are very different from the PE, since their Q values are very close to pure chance.

	MI×χ^2	MI×PE	χ^2×PE
Q	0.71	0.55	0.45

Table 2: The probability Q of having 2 trigrams with the same rank order for different statistical measures.

5.3 Comparing Statistical Measures - are they useful?

The use of statistical measures is widespread in NLP but there is no consensus about how good these measures are for describing natural language phenomena. It is not clear what exactly they capture when analysing the data.

In order to evaluate if they would make good predictors for MWEs, we compare the measures distributions for MWEs and non-MWEs. For that we selected as gold standard a set of around 400 MWE candidates annotated by a native speaker[1] as MWEs or not. We then calculated the histograms for the values of MI, χ^2 and PE for the two groups. MI and χ^2 were calculated only for BNC$_f$. Table 3 displays the results of the Kolmogorov-Smirnof test (Press et al., 1992) for these histograms, where the first value is Kolmogorov-Smirnov D value (D∈[0,1] and large D values indicate large differences between distributions) and the second is the significance probability (p) associated to D given the sizes of the data sets, in this case 90 for MWEs and 292 for non-MWEs.

	MI$_{BNC_f}$	$\chi^2_{BNC_f}$	PE$_{Yahoo}$	PE$_{Google}$
D	0.27	0.13	0.27	0.24
p<	0.0001	0.154	0.0001	0.0005

Table 3: Comparison of MI, χ^2 and PE

The surprising result is that there is no statistical significance, at least using the Kolmogorov-Smirnov test, that indicates that being or not an MWE has some effect in the value of the trigram's χ^2. The same does not happen for MI or PE. They do seem to differentiate between MWEs and non-MWEs. As discussed before the statistical significance implies the existence of an

[1] The native speaker is a linguist expert in MWEs.

effect but has very little to say about the intensity of the effect. As in the case of this work our interest is to use the effect to predict MWEs, the intensity is very important. In the figures that follow we show the normalised histograms for MI, χ^2(for the BNC$_f$) and PE (for the case of Yahoo) for MWEs and non-MWEs. The ideal scenario would be to have non overlapping distributions for the two cases, so a simple threshold operation would be enough to distinguish MWEs. This is not the case in any of the plots. Starting from Figure 3 it clearly illustrates the negative result for χ^2 in table 3. The other two distributions show a visible effect in the form of a slight displacement of the distributions to the left for MWEs. In particular for the distribution of PE, the large peak on the right, representing the n-grams whose word order is irrelevant with respect to its occurrence, has an important reduction for MWEs.

The statistical measures discussed here are all different forms of measuring correlations between the component words of MWEs. Therefore, as some types of MWEs may have stronger constraints on word order, we believe that more visible effects can be seen in these measures if we look at their application for individual types of MWEs, which is planned for future work. This will bring an improvement to the power of MWE prediction of these measures.

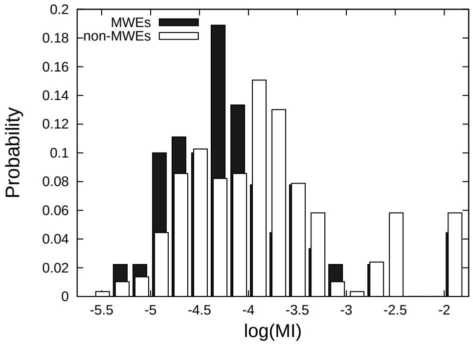


Figure 2: Normalised histograms of MI values for MWEs and non-MWEs in BNC$_f$.

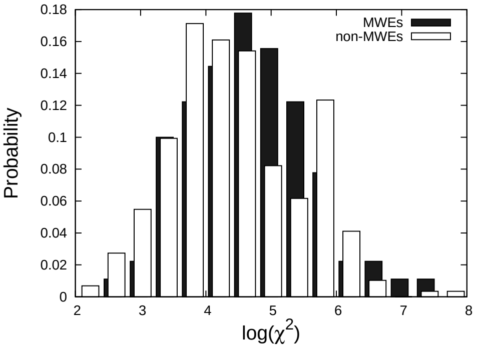


Figure 3: Normalised histograms of χ^2 values for MWEs and non-MWEs in BNC$_f$.

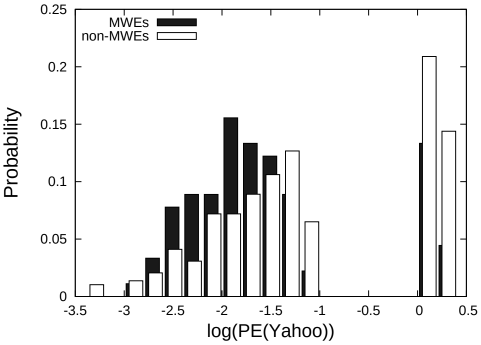


Figure 4: Normalised histograms of PE values for MWEs and non-MWEs in Yahoo.

6 Evaluation of the Extensions to the Grammar

Our ultimate goal is to maximally automate the process of discovering and handling MWEs. With good statistical measures, we are able to distinguish genuine MWE from non-MWEs among the n-gram candidates. However, from the perspective of grammar engineering, even with a good candidate list of MWEs, great effort is still required in order to incorporate such word units into a given grammar automatically and in a precise way.

Zhang et al. (2006) tried a simple "word with spaces" approach. By acquiring new lexical entries for the MWEs candidates validated by the statistical measures, the grammar coverage was shown to improve significantly. However, no further investigation on the parser accuracy was reported there.

Taking a closer look at the MWE candidates proposed, we find that only a small proportion of them can be handled appropriately by the "word with spaces" approach of Zhang et al. (2006). Simply adding new lexical entries for all MWEs can be a workaround for enhancing the parser coverage, but the quality of the parser output is clearly linguistically less interesting.

On the other hand, we also find that a large proportion of MWEs that cannot be correctly handled by the grammar can be covered properly in a constructional way by adding one lexical entry for the head (governing) word of the MWE. For example, the expression *foot the bill* will be correctly handled with a standard head-complement rule, if there is a transitive verb reading for the word *foot* in the lexicon. Some other examples are: *to **put** forward*, *the **good** of*, *in **combination** with*, . . . , where lexical extension to the words in ***bold*** will allow the grammar to cover these MWEs. In this paper, we employ a constructional approach for the acquisition of new lexical entries for the head words of the MWEs.[2]

It is arguable that such an approach may lead to some potential grammar overgeneration, as there is no selectional restriction expressed in the new lexical entry. However, as far as the parsing task is concerned, such overgeneration is not likely to reduce the accuracy of the grammar significantly as we show later in this paper through a thorough evaluation.

6.1 Experimental Setup

With the complete list of 1039 MWE candidates discussed in section 5, we rank each n-gram according to each of the three statistical measures. The average of all the rankings is used as the combined measure of the MWE candidates. Since we are only interested in acquiring new lexical entries for MWEs which are not covered by the grammar, we used the error mining results (Zhang et al., 2006; van Noord, 2004) to only keep those candidates with parsability ≤ 0.1. The top 30 MWE candidates are used in this experiment.

We used simple heuristics in order to extract the head words from these MWEs:

- the n-grams are POS-tagged with an automatic tagger;
- finite verbs in the n-grams are extracted as head words;
- nouns are also extracted if there is no verb in the n-gram.

Occasionally, the tagger errors might introduce wrong head words. However, the lexical type predictor of Zhang and Kordoni (2006) that we used in our experiments did not generate interesting new entries for them in the subsequent steps, and they were thus discarded, as discussed below.

With the 30 MWE candidates, we extracted a sub-corpus from the BNC with 674 sentences which included at least one of these MWEs. The lexical acquisition technique described in Zhang and Kordoni (2006) was used with this sub-corpus in order to acquire new lexical entries for the head words. The lexical acquisition model was trained with the Redwoods treebank (Oepen et al., 2002), following Zhang et al. (2006).

The lexical prediction model predicted for each occurrence of the head words a most plausible lexical type in that context. Only those predictions that occurred 5 times or more were taken into consideration for the generation of the new lexical entries. As a result, we obtained 21 new lexical entries.

These new lexical entries were later merged into the ERG lexicon. To evaluate the grammar performance with and without these new lexical entries, we

1. parsed the sub-corpus with/without new lexical entries and compared the grammar coverage;
2. inspected the parser output manually and evaluated the grammar accuracy.

In parsing the sub-corpus, we used the PET parser (Callmeier, 2001). For the manual eval-

[2] The combination of the "word with space" approach of Zhang et al. (2006) with the constructional approach we propose here is an interesting topic that we want to investigate in future research.

uation of the parser output, we used the treebanking tools of the [incr tsdb()] system (Oepen, 2001).

6.2 Grammar Performance

Table 4 shows that the grammar coverage improved significantly (from 7.1% to 22.7%) with the acquired lexical entries for the head words of the MWEs. This improvement in coverage is largely comparable to the result reported in (Zhang et al., 2006), where the coverage was reported to raise from 5% to 18% with the "word with spaces" approach (see also section 4).

It is also worth mentioning that Zhang et al. (2006) added 373 new lexical entries for a total of 311 MWE candidates, with an average of 1.2 entries per MWE. In our experiment, we achieved a similar coverage improvement with only 21 new entries for 30 different MWE candidates, with an average of 0.7 entries per MWE. This suggests that the lexical entries acquired in our experiment are of much higher linguistic generality.

To evaluate the grammar accuracy, we manually checked the parser outputs for the sentences in the sub-corpus which received at least one analysis from the grammar before and after the lexical extension. Before the lexical extension, 48 sentences are parsed, among which 32 (66.7%) sentences contain at least one correct reading (table 4). After adding the 21 new lexical entries, 153 sentences are parsed, out of which 124 (81.0%) sentences contain at least one correct reading.

Baldwin et al. (2004) reported in an earlier study that for BNC data, about 83% of the sentences covered by the ERG have a correct parse. In our experiment, we observed a much lower accuracy on the sub-corpus of BNC which contains a lot of MWEs. However, after the lexical extension, the accuracy of the grammar recovers to the normal level.

It is also worth noticing that we did not receive a larger average number of analyses per sentence (table 4), as it was largely balanced by the significant increase of sentences covered by the new lexical entries. We also found that the disambiguation model as described by Toutanova et al. (2002) performed reasonably well, and the best analysis is ranked among top-5 for 66% of the cases, and top-10 for 75%.

All of these indicate that our approach of lexical acquisition for head words of MWEs achieves a significant improvement in grammar coverage without damaging the grammar accuracy. Optionally, the grammar developers can check the validity of the lexical entries before they are added into the lexicon. Nonetheless, even a semi-automatic procedure like this can largely reduce the manual work of grammar writers.

7 Conclusions

In this paper we looked at some of the issues involved in the evaluation of the identification of MWEs. In particular we evaluated the use of three statistical measures for automatically identifying MWEs. The results suggest that at least two of them (MI and PE) can distinguish MWEs. In terms of the corpora used, a surprisingly higher level of agreement was found between different corpora (Google and Yahoo) than between two fragments of the same one. This tells us two lessons. First that even though Google and Yahoo were not carefully built to be language corpora their sizes compensate for that making them fairly good samples of language usage. Second, a fraction of a smaller well balanced corpus may not necessarily be as balanced as the whole.

Furthermore, we argued that for precise grammar engineering it is important to perform a careful evaluation of the effects of including automatically acquired MWEs to a grammar. We looked at the evaluation of the effects in coverage, size of the grammar and accuracy of the parses after adding the MWE-candidates. We adopted a compositional approach to the encoding of MWEs, using some heuristics to detect the head of an MWE, and this resulted in a smaller grammar than that by Zhang et al. (2006), still achieving a similar increase in coverage and maintaining a high level of accuracy of parses, comparable to that reported by Baldwin et al. (2004).

The statistical measures are currently only

	item #	parsed #	avg. analysis #	coverage %
ERG	674	48	335.08	7.1%
ERG + MWE	674	153	285.01	22.7%

Table 4: ERG coverage with/without lexical acquisition for the head words of MWEs

used in a preprocessing step to filter the non-MWEs for the lexical type predictor. Alternatively, the statistical outcomes can be incorporated more tightly, i.e. to combine with the lexical type predictor and give confidence scores on the resulting lexical entries. These possibilities will be explored in future work.

References

Timothy Baldwin and Aline Villavicencio. 2002. Extracting the unextractable: A case study on verb-particles. In *Proc. of the 6th Conference on Natural Language Learning (CoNLL-2002)*, Taipei, Taiwan.

Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of dutch. In *Computational Linguistics in The Netherlands 2000*.

Ted Briscoe and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Fifth Conference on Applied Natural Language Processing*, Washington, USA.

Ulrich Callmeier. 2001. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

Stefan Evert and Brigitte Krenn. 2005. Using small random samples for the manual evaluation of statistical association measures. *Computer Speech and Language*, 19(4):450–466.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Ray Jackendoff. 1997. Twistin' the night away. *Language*, 73:534–59.

Jeremy Nicholson and Timothy Baldwin. 2006. Interpretation of compound nominalisations using corpus and web statistics. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 54–61, Sydney, Australia. Association for Computational Linguistics.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*, Taipei.

Stephan Oepen. 2001. [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Darren Pearce. 2002. A comparative evaluation of collocation extraction techniques. In *Third International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands, Spain.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing. Second edition.* Cambridge University Press.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 1–15, Mexico City, Mexico.

Kristina Toutanova, Christoper D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse ranking for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263, Sozopol, Bulgaria.

Leonoor van der Beek. 2005. The extraction of determinerless pps. In *Proceedings of the ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, Colchester, UK.

Gertjan van Noord. 2004. Error mining for wide-coverage grammar engineering. In *Proceedings of* *the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 446–453, Barcelona, Spain, July.

Aline Villavicencio, Francis Bond, Anna Korhonen, and Diana McCarthy. 2005. Introduction to the special issue on multiword expressions: having a crack at a hard nut. *Journal of Computer Speech and Language Processing*, 19(4):365–377.

Aline Villavicencio. 2005. The availability of verb-particle constructions in lexical resources: How much is enough? *Journal of Computer Speech and Language Processing*, 19.

Yi Zhang and Valia Kordoni. 2006. Automated deep lexical acquisition for robust open texts processsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Yi Zhang, Valia Kordoni, Aline Villavicencio, and Marco Idiart. 2006. Automated multiword expression prediction for grammar engineering. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 36–44, Sydney, Australia. Association for Computational Linguistics.

Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles

Kenji Sagae[1] and **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
[2]School of Computer Science, University of Manchester
[3]National Center for Text Mining
{sagae,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We present a data-driven variant of the LR algorithm for dependency parsing, and extend it with a best-first search for probabilistic generalized LR dependency parsing. Parser actions are determined by a classifier, based on features that represent the current state of the parser. We apply this parsing framework to both tracks of the CoNLL 2007 shared task, in each case taking advantage of multiple models trained with different learners. In the multilingual track, we train three LR models for each of the ten languages, and combine the analyses obtained with each individual model with a maximum spanning tree voting scheme. In the domain adaptation track, we use two models to parse unlabeled data in the target domain to supplement the labeled out-of-domain training set, in a scheme similar to one iteration of co-training.

1 Introduction

There are now several approaches for multilingual dependency parsing, as demonstrated in the CoNLL 2006 shared task (Buchholz and Marsi, 2006). The dependency parsing approach presented here extends the existing body of work mainly in four ways:

1. Although *stepwise* [1] dependency parsing has commonly been performed using parsing algorithms designed specifically for this task, such as those described by Nivre (2003) and Yamada and Matsumoto (2003), we show that this can also be done using the well known LR parsing algorithm (Knuth, 1965), providing a connection between current research on shift-reduce dependency parsing and previous parsing work using LR and GLR models;
2. We generalize the standard deterministic stepwise framework to probabilistic parsing, with the use of a best-first search strategy similar to the one employed in constituent parsing by Ratnaparkhi (1997) and later by Sagae and Lavie (2006);
3. We provide additional evidence that the parser ensemble approach proposed by Sagae and Lavie (2006a) can be used to improve parsing accuracy, even when only a single parsing algorithm is used, as long as variation can be obtained, for example, by using different learning techniques or changing parsing direction from *forward* to *backward* (of course, even greater gains may be achieved when different algorithms are used, although this is not pursued here); and, finally,
4. We present a straightforward way to perform parser domain adaptation using unlabeled data in the target domain.

We entered a system based on the approach described in this paper in the CoNLL 2007 shared

[1] *Stepwise* parsing considers each step in a parsing algorithm separately, while *all-pairs* parsing considers entire trees. For a more complete definition, see the CoNLL-X shared task description paper (Buchholz and Marsi, 2006).

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1044–1050,
Prague, June 2007.

task (Nivre et al., 2007), which differed from the 2006 edition by featuring two separate tracks, one in multilingual parsing, and a new track on domain adaptation for dependency parsers. In the multilingual parsing track, participants train dependency parsers using treebanks provided for ten languages: Arabic (Hajic et al., 2004), Basque (Aduriz et al. 2003), Catalan (Martí et al., 2007), Chinese (Chen et al., 2003), Czech (Böhmova et al., 2003), English (Marcus et al., 1993; Johansson and Nugues, 2007), Greek (Prokopidis et al., 2005), Hungarian (Czendes et al., 2005), Italian (Montemagni et al., 2003), and Turkish (Oflazer et al., 2003). In the domain adaptation track, participants were provided with English training data from the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993) converted to dependencies (Johansson and Nugues, 2007) to train parsers to be evaluated on material in the biological (development set) and chemical (test set) domains (Kulick et al., 2004), and optionally on text from the CHILDES database (MacWhinney, 2000; Brown, 1973).

Our system's accuracy was the highest in the domain adaptation track (with labeled attachment score of 81.06%), and only 0.43% below the top scoring system in the multilingual parsing track (our average labeled attachment score over the ten languages was 79.89%). We first describe our approach to multilingual dependency parsing, followed by our approach for domain adaptation. We then provide an analysis of the results obtained with our system, and discuss possible improvements.

2 A Probabilistic LR Approach for Dependency Parsing

Our overall parsing approach uses a best-first probabilistic shift-reduce algorithm based on the LR algorithm (Knuth, 1965). As such, it follows a bottom-up strategy, or *bottom-up-trees*, as defined in Buchholz and Marsi (2006), in contrast to the shift-reduce dependency parsing algorithm described by Nivre (2003), which is a bottom-up/top-down hybrid, or *bottom-up-spans*. It is unclear whether the use of a bottom-up-trees algorithm has any advantage over the use of a bottom-up-spans algorithm (or vice-versa) in practice, but the availability of different algorithms that perform the same parsing task could be advantageous in parser ensembles. The main difference between our parser and a traditional LR parser is that we do not use an LR table derived from an explicit grammar to determine shift/reduce actions. Instead, we use a classifier with features derived from much of the same information contained in an LR table: the top few items on the stack, and the next few items of lookahead in the remaining input string. Additionally, following Sagae and Lavie (2006), we extend the basic deterministic LR algorithm with a best-first search, which results in a parsing strategy similar to generalized LR parsing (Tomita, 1987; 1990), except that we do not perform Tomita's stack-merging operations.

The resulting algorithm is projective, and non-projectivity is handled by pseudo-projective transformations as described in (Nivre and Nilsson, 2005). We use Nivre and Nilsson's *PATH* scheme[2].

For clarity, we first describe the basic variant of the LR algorithm for dependency parsing, which is a deterministic stepwise algorithm. We then show how we extend the deterministic parser into a best-first probabilistic parser.

2.1 Dependency Parsing with a Data-Driven Variant of the LR Algorithm

The two main data structures in the algorithm are a stack *S* and a queue *Q*. *S* holds subtrees of the final dependency tree for an input sentence, and *Q* holds the words in an input sentence. *S* is initialized to be empty, and *Q* is initialized to hold every word in the input in order, so that the first word in the input is in the front of the queue.[3]

The parser performs two main types of actions: *shift* and *reduce*. When a shift action is taken, a word is shifted from the front of *Q*, and placed on the top of *S* (as a tree containing only one node, the word itself). When a reduce action is taken, the

[2] The *PATH* scheme was chosen (even though Nivre and Nilsson report slightly better results with the *HEAD* scheme) because it does not result in a potentially quadratic increase in the number of dependency label types, as observed with the *HEAD* and *HEAD+PATH* schemes. Unfortunately, experiments comparing the use of the different pseudo-projectivity schemes were not performed due to time constraints.

[3] We append a "virtual root" word to the beginning of every sentence, which is used as the head of every word in the dependency structure that does not have a head in the sentence.

two top items in S (s_1 and s_2) are popped, and a new item is pushed onto S. This new item is a tree formed by making the root s_1 of a dependent of the root of s_2, or the root of s_2 a dependent of the root of s_1. Depending on which of these two cases occur, we call the action *reduce-left* or *reduce-right*, according to whether the head of the new tree is to the left or to the right its new dependent. In addition to deciding the direction of a reduce action, the label of the newly formed dependency arc must also be decided.

Parsing terminates successfully when Q is empty (all words in the input have been processed) and S contains only a single tree (the final dependency tree for the input sentence). If Q is empty, S contains two or more items, and no further reduce actions can be taken, parsing terminates and the input is rejected. In such cases, the remaining items in S contain partial analyses for contiguous segments of the input.

2.2 A Probabilistic LR Model for Dependency Parsing

In the traditional LR algorithm, parser states are placed onto the stack, and an LR table is consulted to determine the next parser action. In our case, the parser state is encoded as a set of features derived from the contents of the stack S and queue Q, and the next parser action is determined according to that set of features. In the deterministic case described above, the procedure used for determining parser actions (a classifier, in our case) returns a single action. If, instead, this procedure returns a list of several possible actions with corresponding probabilities, we can then parse with a model similar to the probabilistic LR models described by Briscoe and Carroll (1993), where the probability of a parse tree is the product of the probabilities of each of the actions taken in its derivation.

To find the most probable parse tree according to the probabilistic LR model, we use a best-first strategy. This involves an extension of the deterministic shift-reduce into a best-first shift-reduce algorithm. To describe this extension, we first introduce a new data structure T_i that represents a parser state, which includes a stack S_i, a queue Q_i, and a probability P_i. The deterministic algorithm is a special case of the probabilistic algorithm where we have a single parser state T_0 that contains S_0 and Q_0, and the probability of the parser state is 1. The best-first algorithm, on the other hand, keeps a heap H containing multiple parser states T_0... T_m. These states are ordered in the heap according to their probabilities, which are determined by multiplying the probabilities of each of the parser actions that resulted in that parser state. The heap H is initialized to contain a single parser state T_0, which contains a stack S_0, a queue Q_0 and probability $P_0 = 1.0$. S_0 and Q_0 are initialized in the same way as S and Q in the deterministic algorithm. The best-first algorithm then loops while H is non-empty. At each iteration, first a state $T_{current}$ is popped from the top of H. If $T_{current}$ corresponds to a final state ($Q_{current}$ is empty and $S_{current}$ contains a single item), we return the single item in $S_{current}$ as the dependency structure corresponding to the input sentence. Otherwise, we get a list of parser actions $act_0...act_n$ (with associated probabilities $Pact_0...Pact_n$) corresponding to state $T_{current}$. For each of these parser actions act_j, we create a new parser state T_{new} by applying act_j to $T_{current}$, and set the probability T_{new} to be $P_{new} = P_{currnet} * Pact_j$. Then, T_{new} is inserted into the heap H. Once new states have been inserted onto H for each of the n parser actions, we move on to the next iteration of the algorithm.

3 Multilingual Parsing Experiments

For each of the ten languages for which training data was provided in the multilingual track of the CoNLL 2007 shared task, we trained three LR models as follows. The first LR model for each language uses maximum entropy classification (Berger et al., 1996) to determine possible parser actions and their probabilities[4]. To control overfitting in the MaxEnt models, we used box-type inequality constraints (Kazama and Tsujii, 2003). The second LR model for each language also uses MaxEnt classification, but parsing is performed *backwards*, which is accomplished simply by reversing the input string before parsing starts. Sagae and Lavie (2006a) and Zeman and Žabokrtský (2005) have observed that reversing the direction of stepwise parsers can be beneficial in parser combinations. The third model uses support vector machines[5] (Vapnik, 1995) using the polynomial

[4] Implementation by Yoshimasa Tsuruoka, available at http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/maxent/

[5] Implementation by Taku Kudo, available at http://chasen.org/~taku/software/TinySVM/ and all vs. all was used for multi-class classification.

kernel with degree 2. Probabilities were estimated for SVM outputs using the method described in (Platt, 1999), but accuracy improvements were not observed during development when these estimated probabilities were used instead of simply the single best action given by the classifier (with probability 1.0), so in practice the SVM parsing models we used were deterministic.

At test time, each input sentence is parsed using each of the three LR models, and the three resulting dependency structures are combined according to the maximum-spanning-tree parser combination scheme[6] (Sagae and Lavie, 2006a) where each dependency proposed by each of the models has the same weight (it is possible that one of the more sophisticated weighting schemes proposed by Sagae and Lavie may be more effective, but these were not attempted). The combined dependency tree is the final analysis for the input sentence.

Although it is clear that fine-tuning could provide accuracy improvements for each of the models in each language, the same set of meta-parameters and features were used for all of the ten languages, due to time constraints during system development. The features used were[7]:

- For the subtrees in *S(1)* and *S(2)*
  - the number of children of the root word of the subtrees;
  - the number of children of the root word of the subtree to the right of the root word;
  - the number of children of the root word of the subtree to the left of the root word;
  - the POS tag and DEPREL of the rightmost and leftmost children;
- The POS tag of the word immediately to the right of the root word of *S(2)*;
- The POS tag of the word immediately to the left of *S(1)*;
- The previous parser action;
- The features listed for the root words of the subtrees in table 1.

In addition, the MaxEnt models also used selected combinations of these features. The classes used to represent parser actions were designed to encode all aspects of an action (shift vs. reduce, right vs. left, and dependency label) simultaneously.

Results for each of the ten languages are shown in table 2 as labeled and unlabeled attachment scores, along with the average labeled attachment score and highest labeled attachment score for all participants in the shared task. Our results shown in boldface were among the top three scores for those particular languages (five out of the ten languages).

	S(1)	*S(2)*	*S(3)*	*Q(0)*	*Q(1)*	*Q(3)*
WORD	x	x	x	x	x	
LEMMA	x	x		x		
POS	x	x	x	x	x	x
CPOS	x	x		x		
FEATS	x	x		x		

Table 1: Additional features.

Language	LAS	UAS	Avg LAS	Top LAS
Arabic	74.71	84.04	68.34	76.52
Basque	74.64	**81.19**	68.06	76.94
Catalan	**88.16**	**93.34**	79.85	88.70
Chinese	**84.69**	**88.94**	76.59	84.69
Czech	74.83	81.27	70.12	80.19
English	**89.01**	**89.87**	80.95	89.61
Greek	73.58	80.37	70.22	76.31
Hungarian	**79.53**	**83.51**	71.49	80.27
Italian	**83.91**	**87.68**	78.06	84.40
Turkish	75.91	82.72	70.06	79.81
ALL	79.90	85.29	65.50	80.32

Table 2: Multilingual results.

4 Domain Adaptation Experiments

In a similar way as we used multiple LR models in the multilingual track, in the domain adaptation track we first trained two LR models on the out-of-

[6] Each dependency tree is deprojectivized before the combination occurs.

[7] *S(n)* denotes the *nth* item from the top of the stack (where *S(1)* is the item on top of the stack), and *Q(n)* denotes the *nth* item in the queue. For a description of the features names in capital letters, see the shared task description (Nivre et al., 2007).

domain labeled training data. The first was a forward MaxEnt model, and the second was a backward SVM model. We used these two models to perform a procedure similar to a single iteration of co-training, except that selection of the newly (automatically) produced training instances was done by selecting sentences for which the two models produced identical analyses. On the development data we verified that sentences for which there was perfect agreement between the two models had labeled attachment score just above 90 on average, even though each of the models had accuracy between 78 and 79 over the entire development set.

Our approach was as follows:

1. We trained the forward MaxEnt and backward SVM models using the out-of-domain labeled training data;
2. We then used each of the models to parse the first two of the three sets of domain-specific unlabeled data that were provided (we did not use the larger third set)
3. We compared the output for the two models, and selected only identical analyses that were produced by each of the two separate models;
4. We added those analyses (about 200k words in the test domain) to the original (out-of-domain) labeled training set;
5. We retrained the forward MaxEnt model with the new larger training set; and finally
6. We used this model to parse the test data.

Following this procedure we obtained a labeled attachment score of 81.06, and unlabeled attachment score of 83.42, both the highest scores for this track. This was done without the use of any additional resources (closed track), but these results are also higher than the top score for the open track, where the use of certain additional resources was allowed. See (Nivre et al., 2007).

5 Analysis and Discussion

One of the main assumptions in our use of different models based on the same algorithm is that while the output generated by those models may often differ, agreement between the models is an indication of correctness. In our domain adaptation approach, this was clearly true. In fact, the approach would not have worked if this assumption was false. Experiments on the development set were encouraging. As stated before, when the parsers agreed, labeled attachment score was over 90, even though the score of each model alone was lower than 79. The domain-adapted parser had a score of 82.1, a significant improvement. Interestingly, the ensemble used in the multilingual track also produced good results on the development set for the domain adaptation data, without the use of the unlabeled data at all, with a score of 81.9 (although the ensemble is more expensive to run).

The different models used in each track were distinct in a few ways: (1) direction (forward or backward); (2) learner (MaxEnt or SVM); and (3) search strategy (best-first or deterministic). Of those differences, the first one is particularly interesting in single-stack shift-reduce models, as ours. In these models, the context to each side of a (potential) dependency differs in a fundamental way. To one side, we have tokens that have already been processed and are already in subtrees, and to the other side we simply have a look-ahead of the remaining input sentence. This way, the context of the same dependency in a forward parser may differ significantly from the context of the same dependency in a backward parser. Interestingly, the accuracy scores of the MaxEnt backward models were found to be generally just below the accuracy of their corresponding forward models when tested on development data, with two exceptions: Hungarian and Turkish. In Hungarian, the accuracy scores produced by the forward and backward MaxEnt LR models were not significantly different, with both labeled attachment scores at about 77.3 (the SVM model score was 76.1, and the final combination score on development data was 79.3). In Turkish, however, the backward score was significantly higher than the forward score, 75.0 and 72.3, respectively. The forward SVM score was 73.1, and the combined score was 75.8. In experiments performed after the official submission of results, we evaluated a *backward* SVM model (which was trained after submission) on the same development set, and found it to be significantly more accurate than the forward model, with a score of 75.7. Adding that score to the combination raised the combination score to 77.9 (a large improvement from 75.8). The likely reason for this difference is that over 80% of the dependencies in the Turkish data set have the head to the right of the dependent, while only less than 4% have the head to the left. This means that the backward model builds much more partial structure in the stack as it consumes input tokens, while the forward model must consume most tokens before it starts making attachments. In other words, context in general in the backward model has more structure, and attachments are made while there are still look-ahead tokens, while the opposite is generally true in the forward model.

6 Conclusion

Our results demonstrate the effectiveness of even small ensembles of parsers that are relatively similar (using the same features and the same algorithm). There are several possible extensions and improvements to the approach we have described. For example, in section 3 we mention the use of different weighting schemes in dependency voting. We list additional ideas that were not attempted due to time constraints, but that are likely to produce improved results.

One of the simplest improvements to our approach is simply to train more models with no other changes to our set-up. As mentioned in section 5, the addition of a backward SVM model did improve accuracy on the Turkish set significantly, and it is likely that improvements would also be obtained in other languages. In addition, other learning approaches, such as memory-based language processing (Daelemans and Van den Bosch, 2005), could be used. A drawback of adding more models that became obvious in our experiments was the increased cost of both training (for example, the SVM parsers we used required significantly longer to train than the MaxEnt parsers) and run-time (parsing with MBL models can be several times slower than with MaxEnt, or even SVM). A similar idea that may be more effective, but requires more effort, is to add parsers based on different approaches. For example, using MSTParser (McDonald and Pereira, 2005), a large-margin all-pairs parser, in our domain adaptation procedure results in significantly improved accuracy (83.2 LAS). Of course, the use of different approaches used by different groups in the CoNLL 2006 and 2007 shared tasks represents great opportunity for parser ensembles.

Acknowledgements

We thank the shared task organizers and treebank providers. We also thank the reviewers for their comments and suggestions, and Yusuke Miyao for insightful discussions. This work was supported in part by Grant-in-Aid for Specially Promoted Research 18002007.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

A. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to naturallanguage processing. *Computational Linguistics*, 22(1):39–71.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia and M. Oronoz. 2003. Construction of a Basque Dependency Treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajic, E. Hajicová and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, 103–127.

E. Briscoe and J. Carroll. 1993. Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. In *Computational Linguistics*, 19(1), pages 25-59.

R. Brown. 1973. *A First Language: The Early Stages*. Harvard University Press.

S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proc. of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. New York, NY.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang and Z. Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé (2003), chapter 13, pages 231–248.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

W. Daelemans and A. Van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press.

J. Hajic, O. Smrz, P. Zemánek, J. Snaidauf and E. Beska. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA).*

J. Kazama, and J. Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of EMNLP 2003.*

D. Knuth. 1965. On the translation of languages from left to right, *Information and Control* 8, 607-639.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL).*

B. MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Lawrence Erlbaum.

R. McDonald, K.Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez and M. Bertran. 2007. CESS-ECE: A Multilingual and Multilevel Annotated Corpus. Available for download from: http://www.lsi.upc.edu/~mbertran/cess-ece/.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

J. Nivre. 2003. An efficient algorithm for dependency parsing. In *Proc. of the Eighth International Workshop on Parsing Technologies (IWPT'03).* Nancy, France.

J. Nivre, and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL),* 99-106. Ann Arbor, MI.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

J. Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classiers*, MIT Press.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek depen- dency treebank. In Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT), pages 149–160.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing.* Providence, RI

K. Sagae, and A. Lavie. 2006. A best-first probabilistic shift-reduce parser. *Proceedings of the 43rd Meeting of the Association for Computational Linguistics - posters (ACL'06).* Sydney, Australia.

K. Sagae, and A. Lavie. 2006a. Parser combination by reparsing. *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics - short papers (HLT-NAACL'06).* New York, NY.

M. Tomita. 1987. An efficient augmented context-free parsing algorithm. *Computational Linguistics*, 13:31–46.

M. Tomita. 1990. The generalized LR parser/compiler - version 8.4. In *Proceedings of the International Conference on Computational Linguistics (COLING'90)*, pages 59–63. Helsinki, Finland.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag.

H. Yamada, and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT'03).* Nancy, France.

D. Zeman, Z. Žabokrtský. 2005. Improving Parsing Accuracy by Combining Diverse Dependency Parsers. In *Proceedings of the International Workshop on Parsing Technologies (IWPT 2005).* Vancouver, British Columbia.

Frustratingly Hard Domain Adaptation for Dependency Parsing

Mark Dredze[1] and **John Blitzer**[1] and **Partha Pratim Talukdar**[1] and
Kuzman Ganchev[1] and **João V. Graça**[2] and **Fernando Pereira**[1]

[1]CIS Dept., University of Pennsylvania, Philadelphia, PA 19104
{mdredze|blitzer|partha|kuzman|pereira}@seas.upenn.edu

[2]L^{2}F – INESC-ID Lisboa/IST, Rua Alves Redol 9, 1000-029, Lisboa, Portugal
javg@l2f.inesc-id.pt

Abstract

We describe some challenges of adaptation in the 2007 CoNLL Shared Task on Domain Adaptation. Our error analysis for this task suggests that a primary source of error is differences in annotation guidelines between treebanks. Our suspicions are supported by the observation that no team was able to improve target domain performance substantially over a state of the art baseline.

1 Introduction

Dependency parsing, an important NLP task, can be done with high levels of accuracy. However, adapting parsers to new domains *without* target domain labeled training data remains an open problem. This paper outlines our participation in the 2007 CoNLL Shared Task on Domain Adaptation (Nivre et al., 2007). The goal was to adapt a parser trained on a single source domain to a new target domain using only unlabeled data. We were given around 15K sentences of labeled text from the Wall Street Journal (WSJ) (Marcus et al., 1993; Johansson and Nugues, 2007) as well as 200K unlabeled sentences. The development data was 200 sentences of labeled biomedical oncology text (BIO, the ONCO portion of the Penn Biomedical Treebank), as well as 200K unlabeled sentences (Kulick et al., 2004). The two test domains were a collection of medline chemistry abstracts (pchem, the CYP portion of the Penn Biomedical Treebank) and the Child Language Data Exchange System corpus (CHILDES) (MacWhinney, 2000; Brown, 1973). We used the second order two stage parser and edge labeler of McDonald et al. (2006), which achieved top results in the 2006 CoNLL-X shared task. Preliminary experiments indicated that the edge labeler was fairly robust to domain adaptation, lowering accuracy by 3% in the development domain as opposed to 2% in the source, so we focused on unlabeled dependency parsing.

Our system did well, officially coming in 3rd place out of 12 teams and within 1% of the top system (Table 1). [1] In unlabeled parsing, we scored 1st and 2nd on CHILDES and pchem respectively. However, our results were obtained *without adaptation*. Given our position in the ranking, this suggests that no team was able to significantly improve performance on either test domain beyond that of a state-of-the-art parser.

After much effort in developing adaptation methods, it is critical to understand the causes of these negative results. In what follows, we provide an error analysis that attributes domain loss for this task to a difference in annotation guidelines between domains. We then overview our attempts to improve adaptation. While we were able to show limited adaptation on reduced training data or with first-order features, no modifications improved parsing with all the training data and second-order features.

2 Parsing Challenges

We begin with an error analysis for adaptation between WSJ and BIO. We divided the available WSJ data into a train and test set, trained a parser on the train set and compared errors on the test set and BIO. Accuracy dropped from 90% on WSJ to 84% on BIO. We then computed the fraction of errors involving each POS tag. For the most common

[1] While only 8 teams participated in the closed track with us, our score beat all of the teams in the open track.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1051–1055,
Prague, June 2007. ©2007 Association for Computational Linguistics

	pchem l	*pchem ul*	*childes ul*	*bio ul*
Ours	80.22	83.38	61.37	83.93
Best	81.06	83.42	61.37	-
Mean	73.03	76.42	57.89	-
Rank	3rd	2nd	1st	-

Table 1: Official labeled (l) and other unlabeled (ul) submitted results for the two test domains (pchem and childes) and development data accuracy (bio). The parser was trained on the provided WSJ data.

POS types, the loss (difference in source and target error) was: verbs (2%), conjunctions (5%), digits (23%), prepositions (4%), adjectives (3%), determiners (4%) and nouns (9%). [2] Two POS types stand out: digits and nouns. Digits are less than 4% of the tokens in BIO. Errors result from the BIO annotations for long sequences of digits which do not appear in WSJ. Since these annotations are new with respect to the WSJ guidelines, it is impossible to parse these without injecting knowledge of the annotation guidelines. [3] Nouns are far more common, comprising 33% of BIO and 30% of WSJ tokens, the most popular POS tag by far. Additionally, other POS types listed above (adjectives, prepositions, determiners, conjunctions) often attach to nouns. To confirm that nouns were problematic, we modified a first-order parser (no second order features) by adding a feature indicating correct noun-noun edges, forcing the parser to predict these edges correctly. Adaptation performance rose on BIO from 78% without the feature to 87% with the feature. This indicates that most of the loss comes from missing these edges.

The primary problem for nouns is the difference between structures in each domain. The annotation guidelines for the Penn Treebank flattened noun phrases to simplify annotation (Marcus et al., 1993), so there is no complex structure to NPs. Kübler (2006) showed that it is difficult to compare the Penn Treebank to other treebanks with more complex noun structures, such as BIO. Consider the WSJ phrase “the New York State Insurance Department”. The annotation indicates a flat structure, where every token is headed by “Department”. In contrast, a similar BIO phrase has a very different structure, pursuant to the BIO guidelines. For “the detoxication enzyme glutathione transferase P1-1”, “enzyme” is the head of the NP, “P1-1” is the head of “transferase”, and “transferase” is the head of “glutathione”. Since the guidelines differ, we observe *no* corresponding structure in the WSJ. It is telling that the parser labels this BIO example by attaching every token to the final proper noun “P1-1”, exactly as the WSJ guidelines indicate. Unlabeled data cannot indicate that BIO uses a different standard.

Another problem concerns appositives. For example, the phrase “Howard Mosher, president and chief executive officer,” has “Mosher” as the head of “Howard” and of the appositive NP delimited by commas. While similar constructions occur in BIO, there are no commas to indicate this. An example is the above BIO NP, in which the phrase “glutathione transferase P1-1” is an appositive indicating which “enzyme” is meant. However, since there are no commas, the parser thinks “P1-1” is the head. However, there are not many right to left attaching nouns.

In addition to a change in the annotation guidelines for NPs, we observed an important difference in the distribution of POS tags. NN tags were almost twice as likely in the BIO domain (14% in WSJ and 25% in BIO). NNP tags, which are close to 10% of the tags in WSJ, are nonexistent in BIO (.24%). The cause for this is clear when the annotation guidelines are considered. The proper nouns in WSJ are names of companies, people and places, while in BIO they are names of genes, proteins and chemicals. However, for BIO these nouns are labeled NN instead of NNP. This decision effectively removes NNP from the BIO domain and renders all features that depend on the NNP tag ineffective. In our above BIO NP example, all nouns are labeled NN, whereas the WSJ example contains NNP tags. The largest tri-gram differences involve nouns, such as *NN-NN-NN*, *NNP-NNP-NNP*, *NN-IN-NN*, and *IN-NN-NN*. However, when we examine the coarse POS tags, which do not distinguish between nouns, these differences disappear. This indicates that while the overall distribution of POS tags is similar between the domains, the fine grained tags differ. These fine grained tags provide more information than coarse tags; experiments that removed fine grained tags

[2] We measured these drops on several other dependency parsers and found similar results.

[3] For example, the phrase “(R = 28% (10/26); K=10% (3/29); chi2 test: p=0.014).”

hurt WSJ performance but did not affect BIO.

Finally, we examined the effect of unknown words. Not surprisingly, the most significant differences in error rates concerned dependencies between words of which one or both were unknown to the parser. For two words that were seen in the training data loss was 4%, for a single unknown word loss was 15%, and 26% when both words were unknown. Both words were unknown only 5% of the time in BIO, while one of the words being unknown was more common, reflecting 27% of decisions. Upon further investigation, the majority of unknown words were nouns, which indicates that unknown word errors were caused by the problems discussed above.

Recent theoretical work on domain adaptation (Ben-David et al., 2006) attributes adaptation loss to two sources: the difference in the distribution between domains and the difference in labeling functions. Adaptation techniques focus on the former since it is impossible to determine the latter without knowledge of the labeling function. In parsing adaptation, the former corresponds to a difference between the features seen in each domain, such as new words in the target domain. The decision function corresponds to differences between annotation guidelines between two domains. Our error analysis suggests that the primary cause of loss from adaptation is from differences in the annotation guidelines themselves. Therefore, significant improvements cannot be made without specific knowledge of the target domain's annotation standards. No amount of source training data can help if no relevant structure exists in the data. Given the results for the domain adaptation track, it appears no team successfully adapted a state-of-the-art parser.

3 Adaptation Approaches

We survey the main approaches we explored for this task. While some of these approaches provided a modest performance boost to a simple parser (limited data and first-order features), no method added any performance to our best parser (all data and second-order features).

3.1 Features

A natural approach to improving parsing is to modify the feature set, both by removing features less likely to transfer and by adding features that are more likely to transfer. We began with the first approach and removed a large number of features that we believed transfered poorly, such as most features for noun-noun edges. We obtained a small improvement in BIO performance on limited data only. We then added several different types of features, specifically designed to improve noun phrase constructions, such as features based on the lexical position of nouns (common position in NPs), frequency of occurrence, and NP chunking information. For example, trained on in-domain data, nouns that occur more often tend to be heads. However, none of these features transfered between domains.

A final type of feature we added was based on the behavior of nouns, adjectives and verbs in each domain. We constructed a feature representation of words based on adjacent POS and words and clustered words using an algorithm similar to that of Saul and Pereira (1997). For example, our clustering algorithm grouped first names in one group and measurements in another. We then added the cluster membership as a lexical feature to the parser. None of the resulting features helped adaptation.

3.2 Diversity

Training diversity may be an effective source for adaptation. We began by adding information from multiple different parsers, which has been shown to improve in-domain parsing. We added features indicating when an edge was predicted by another parser and if an edge crossed a predicted edge, as well as conjunctions with edge types. This failed to improve BIO accuracy since these features were less reliable at test time. Next, we tried instance bagging (Breiman, 1996) to generate some diversity among parsers. We selected with replacement 2000 training examples from the training data and trained three parsers. Each parser then tagged the remaining 13K sentences, yielding 39K parsed sentences. We then shuffled these sentences and trained a final parser. This failed to improve performance, possibly because of conflicting annotations or because of lack of sufficient diversity. To address conflicting annotations, we added slack variables to the MIRA learning algorithm (Crammer et al., 2006) used to train the parsers, without success. We measured diversity by comparing the parses of each model. The difference in annotation agreement between the three instance bagging parsers was about half the difference between these parsers and the gold annotations. While we believe this is not enough diversity, it was not feasible to repeat our experiment with a large number of parsers.

3.3 Target Focused Learning

Another approach to adaptation is to favor training examples that are *similar to* the target. We first modified the weight given by the parser to each training sentence based on the similarity of the sentence to target domain sentences. This can be done by modifying the loss to limit updates in cases where the sentence does not reflect the target domain. We tried a number of criteria to weigh sentences without success, including sentence length and number of verbs. Next, we trained a discriminative model on the provided unlabeled data to predict the domain of each sentence based on POS n-grams in the sentence. Training sentences with a higher probability of being in the target domain received higher weights, also without success. Further experiments showed that any decrease in training data hurt parser performance. It would seem that the parser has no difficulty learning important training sentences in the presence of unimportant training examples.

A related idea focused on words, weighing highly tokens that appeared frequently in the target domain. We scaled the loss associated with a token by a factor proportional to its frequency in the target domain. We found certain scaling techniques obtained tiny improvements on the target domain that, while significant compared to competition results, are not statistically significant. We also attempted a similar approach on the feature level. A very predictive source domain feature is not useful if it does not appear in the target domain. However, limiting the feature space to target domain features had no effect. Instead, we scaled each feature's value by a factor proportional to its frequency in the target domain and trained the parser on these scaled feature values. We obtained small improvements on small amounts of training data.

4 Future Directions

Given our pessimistic analysis and the long list of failed methods, one may wonder if parser adaptation is possible at all. We believe that it is. First, there may be room for adaptation with our domains if a common annotation scheme is used. Second, we have stressed that typical adaptation, modifying a model trained on the source domain, will fail but there may be unsupervised parsing techniques that improve performance after adaptation, such as a rule based NP parser for BIO based on knowledge of the annotations. However, this approach is unsatisfying as it does not allow general purpose adaptation.

5 Acknowledgments

We thank Joel Wallenberg and Nikhil Dinesh for their informative and helpful linguistic expertise, Kevin Lerman for his edge labeler code, and Koby Crammer for helpful conversations. Dredze is supported by a NDSEG fellowship; Ganchev and Talukdar by NSF ITR EIA-0205448; and Blitzer by DARPA under Contract No. NBCHD03001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA or the Department of Interior-National Business Center (DOI-NBC).

References

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *NIPS*.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

R. Brown. 1973. *A First Language: The Early Stages*. Harvard University Press.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, Mar.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

Sandra Kübler. 2006. How do treebank annotation schemes influence parsing results? or how not to compare apples and oranges. In *RANLP*.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

B. MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency parsing with a two-stage discriminative parser. In *Conference on Natural Language Learning (CoNLL)*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Lawrence Saul and Fernando Pereira. 1997. Aggregate and mixed-order markov models for statistical language modeling. In *EMNLP*.

Crystal: Analyzing Predictive Opinions on the Web

Soo-Min Kim and Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
{skim,hovy}@ISI.EDU

Abstract

In this paper, we present an election prediction system (*Crystal*) based on web users' opinions posted on an election prediction website. Given a prediction message, *Crystal* first identifies which party the message predicts to win and then aggregates prediction analysis results of a large amount of opinions to project the election results. We collect past election prediction messages from the Web and automatically build a gold standard. We focus on capturing lexical patterns that people frequently use when they express their predictive opinions about a coming election. To predict election results, we apply SVM-based supervised learning. To improve performance, we propose a novel technique which generalizes n-gram feature patterns. Experimental results show that *Crystal* significantly outperforms several baselines as well as a non-generalized n-gram approach. *Crystal* predicts future elections with 81.68% accuracy.

1 Introduction

As a growing number of people use the Web as a medium for expressing their opinions, the Web is becoming a rich source of various opinions in the form of product reviews, travel advice, social issue discussions, consumer complaints, stock market predictions, real estate market predictions, etc.

At least two categories of opinions can be identified. One consists of opinions such as "I like/dislike it", and the other consists of opinions like "It is likely/unlikely to happen." We call the first category ***Judgment Opinions*** and the second (those discussing the future) ***Predictive Opinions***. Judgment opinions express positive or negative sentiment about a topic such as, for example, reviews about cameras, movies, books, or hotels, and discussions about topics like abortion and war. In contrast, predictive opinions express a person's opinion about the future of a topic or event such as the housing market, a popular sports match, and national election, based on his or her belief and knowledge.

Due to the different nature of these two categories of opinion, each has different valences. Judgment opinions have core valences of *positive* and *negative*. For example, "liking a product" and "supporting abortion" have the valence "positive" toward each topic (namely "a product" and "abortion"). Predictive opinions have the core valence of *likely* or *unlikely* predicated on the event. For example, a sentence "Housing prices will go down soon" carries the valence of "likely" for the event of "housing prices go down".

The two types of opinions can co-appear. The sentence "I like Democrats but I think they are not likely to win considering the war issue" contains both types of opinion: "positive" valence towards Democrats and "unlikely" valence towards the event of "Democrats wins". In order to accurately identify and analyze each type of opinion, different approaches are desirable.

Note that our work is different from predictive data mining which models a data mining system using statistical approaches in order to forecast the future or trace a pattern of interest (Rickel and Porter, 1997; Rodionov and Martin, 1996). Example domains of predictive data mining include earthquake prediction, air temperature prediction, foreign exchange prediction, and energy price predic-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1056–1064, Prague, June 2007. ©2007 Association for Computational Linguistics

tion. However, predictive data mining is only feasible when a large amount of structured numerical data (e.g., in a database) is available. Unlike this research area which analyzes numeric values, our study mines unstructured text using NLP techniques and it can potentially extend the reach of numeric techniques.

Despite the vast amount of predictive opinions and their potential applications such as identification and analysis of people's opinions about the real estate market or a specific country's economic future, studies on predictive opinions have been neglected in Computational Linguistics, where most previous work focuses on judgment opinions (see Section 2). In this paper, we concentrate on identifying predictive opinion with its valence.

Among many prediction domains on the Web, we focus on election prediction and introduce *Crystal*, a system to predict election results using the public's written viewpoints. To build our system, we collect opinions about past elections posted on an election prediction project website before the election day, and build a corpus[1]. We then use this corpus to train our system for analyzing predictive opinion messages and, using this, to predict the election outcome. Due to the availability of actual results of the past elections, we can not only evaluate how accurately *Crystal* analyzes prediction messages (by checking agreement with the gold standard), but also objectively measure the prediction accuracy of our system.

The main contributions of this work are as follows:

- an NLP technique for analyzing predictive opinions in the electoral domain;
- a method of automatically building a corpus of predictive opinions for a supervised learning approach; and
- a feature generalization technique that outperforms all the baselines on the task of identifying a predicted winning party given a predictive opinion.

The rest of this paper is structured as follows. Section 2 surveys previous work. Section 3 formally defines our task and describes our data set. Section 4 describes our system *Crystal* with proposed feature generalization algorithm. Section 5 reports empirical evidence that *Crystal* outperforms several baseline systems. Finally, Section 6 concludes with a description of the impact of this work.

2 Related Work

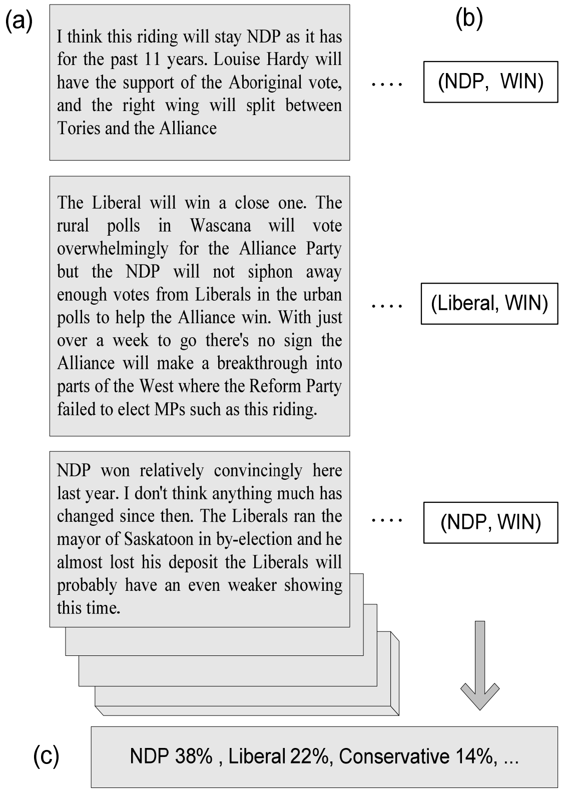
This work is closely related to opinion analysis and text classification. Most research on opinion analysis in computational linguistics has focused on sentiment analysis, subjectivity detection, and review mining. Pang et al. (2002) and Turney (2002) classified sentiment polarity of reviews at the document level. Wiebe et al. (1999) classified sentence level subjectivity using syntactic classes such as adjectives, pronouns and modal verbs as features. Riloff and Wiebe (2003) extracted subjective expressions from sentences using a bootstrapping pattern learning process. Wiebe et. al (2004) and Riloff et. al (2005) adopted pattern learning with lexical feature generalization for subjective expression detection. Dave et. al (2003) and Jindal and Liu (2006) also learned patterns of opinion expression in product reviews. Yu and Hatzivassiloglou (2003) identified the polarity of opinion sentences using semantically oriented words. These techniques were applied and examined in different domains, such as customer reviews (Hu and Liu 2004; Popescu et al., 2005) and news articles (Kim and Hovy, 2004; Wilson et al., 2005).

In text classification, systems typically use bag-of-words models, mostly with supervised learning algorithms using Naive Bayes or Support Vector Machines (Joachims, 1998) to classify documents into several categories such as sports, art, politics, and religion. Liu et al. (2004) and Gliozzo et al. (2005) address the difficulty of obtaining training corpora for supervised learning and propose unsupervised learning approaches. Another recent related classification task focuses on academic and commercial efforts to detect email spam messages. For an SVM-based approach, see (Drucker et al., 1999). In our study, we explore the use of generalized lexical features for predictive opinion analysis and compare it with the bag-of-words approach.

3 Modeling Prediction

In this section, we define the task of analyzing predictive opinions in the electoral domain.

[1] The resulting corpus is available at
http://www.isi.edu/ ~skim/Download/Data/predictive.htm



I think this riding will stay NDP as it has for the past 11 years. Louise Hardy will have the support of the Aboriginal vote, and the right wing will split between Tories and the Alliance

The Liberal will win a close one. The rural polls in Wascana will vote overwhelmingly for the Alliance Party but the NDP will not siphon away enough votes from Liberals in the urban polls to help the Alliance win. With just over a week to go there's no sign the Alliance will make a breakthrough into parts of the West where the Reform Party failed to elect MPs such as this riding.

NDP won relatively convincingly here last year. I don't think anything much has changed since then. The Liberals ran the mayor of Saskatoon in by-election and he almost lost his deposit the Liberals will probably have an even weaker showing this time.

Figure 1. Our election prediction system. Public opinions are collected from message boards (a) and our system determines for each the election prediction 'Party' and 'Valence' (b). The output of the system is a prediction of the election outcome (c).

3.1 Task Definition

We model predictive opinions in an election as follows:

$$ElectionPredictionOpinion = (Party, Valence)$$

where *Party* is a political party running for an election (e.g., Democrats and Republicans) and *Valence* is the valence of a predictive opinion which can be either *"likely to win"* (***WIN***) or *"unlikely to win"* (***LOSE***). Values for *Party* vary depending on in which year (e.g., 1996 and 2006) and where an election takes place (e.g., United States, France, or Japan). The unit of a predictive opinion is an unstructured textual document such as an article in a personal blog or a message posted on a news group discussion board about the topic of "Which party do you think will win/lose in this election?".

Message text	Predicted winning party	Riding	Year
...	...	...	...
Message_1457	Party_3	Riding_206	2004
Message_1458	Party_2	Riding_206	2004
Message_1459	Party_2	Riding_189	2006
Message_1460	Party_1	Riding_189	2006
Message_1461	Party_2	Riding_189	2006
Message_1462	Party_1	Riding_46	2006
...	...	...	...

Table 1. A snapshot of the processed data

Riding name	Party	Candidate name
Blackstrap	NDP	Noreen Johns
	Liberal	J. Wayne Zimmer
	PC	Lynne Yelich

Table 2. An example of our Party-Candidate listing for a riding (PC: Progressive Conservative)

Figure 1 illustrates an overview of our election prediction system *Crystal* in action. Given each document posted on blogs or message boards (e.g., www.election prediction.org) as seen in Figure 1.a, a system can determine a *Party* that the author of a document thinks to win or lose (*Valence*), Figure 1.b. For the example document starting with the sentence "I think this riding will stay NDP as it has for the past 11 years." in Figure 1.a, our predictive opinion analysis system aims to recognize NDP as *Party* and *WIN* as *Valence*. After aggregating the predictive opinion analysis results of all documents, we project the election results in Figure 1.c. The following section describes how we obtain our data set and the subsequent sections describe *Crystal*.

3.2 Automatically Labeled Data

We collected messages posted on an election prediction project page, www.electionprediction. org. The website contains various election prediction projects (e.g., provincial election, federal election, and general election) of different countries (e.g., Canada and United Kingdom) from 1999 to 2006. For our data set, we downloaded Canadian federal election prediction data for 2004 and 2006. The Canadian federal electoral system is based on 308 ridings (electoral districts). The website contains 308 separate html files of messages corresponding to the 308 ridings for different years. In total, we collected 4858 and 4680 messages for the 2004 and 2006 federal elections respectively. On average, a message consists of 98.8 words.

To train and evaluate our system, we require a gold standard for each message (i.e., which party does an author of a message predict to win?). One option is to hire human annotators to build the gold standard. Instead, we used an online party logo image file that the author of each message already labeled for the message. Note that authors only select parties they think will win, which means our gold standard only contains a party with *WIN* valence of each message. However, we leverage this information to build a system which is able to determine a party even with *LOSE* valence. We describe this idea in detail in Section 4.

Finally, we pre-processed the data by converting the downloaded html source files into a structured format with the following fields: *message, party, riding*, and *year*, where *message* is a text, *party* is a winning party predicted in the text, *riding* is one of the 308 ridings, and *year* is either 2004 or 2006. Table 1 shows a snapshot of the processed data set that we used for our system training and evaluation. An additional piece of information consisting of a candidate's name for each party for each riding was also stored in our data set. With this information, the system can infer opinions about a party based on opinions about candidates who run for the party. Table 2 shows an example of a riding.

4 Analyzing Predictions

In this section we describe *Crystal*. One simple approach could be a system (see NGR system in Section 5) trained by a machine learning technique using n-gram features and classifying a message into multiple classes (e.g., NDP, Liberal, or Progressive). However, we develop a more sophisticated algorithm and compare its result with several baselines, including the simple n-gram method[2]. Experimental results in Section 5 show that *Crystal* outperforms all the baselines.

Our approach consists of three steps: feature generalization, classification using SVMs, and SVM result integration[3]. *Crystal* generates generalized sentences in the feature generalization step. Then it classifies each sentence using generalized lexical features in order to determine *Valence* of *Party* in a sentence. Finally, it combines results of sentences to determine *Valence* and *Party* of a message. Note that the classification using SVM is an intermediate step conducting a binary classification (i.e., WIN or LOSE) for the final multi-class classification in result integration. The following sections describe each step.

```
for each message M with a party that M predicts to win, $P_w$
  for each sentence $S_i$ in a message M
    for each party $P_j$ in $S_i$
      valence $V_j$ = +1 if $P_j = P_w$
      valence $V_j$ = -1  Otherwise
      Generate $S'_{ij}$ by substituting $P_j$ with
        **PARTY**
      and all other parties in $S_i$ with **OTHER**
      Return ($P_j$, $V_j$, $S'_{ij}$)
```

Table 3. Feature generalization algorithm

4.1 Feature Generalization

In the feature generalization step, we generalize patterns of words used in predictive opinions. For example, instead of using three different trigrams like "Liberals will win", "NDP will win", and "Conservatives will win", we generalize these to "PARTY will win". The assumption is that the generalized patterns can represent better the relationship among Party, Valence, and words surrounding Party (e.g., will win) than pure lexical patterns. For this algorithm, we first substitute a candidate's name (both the first name and the last name) with the political party name that the candidate belongs to (see Table 2). We then break each message into sentences[4].

Table 3 outlines the feature generalization algorithm. Here, our approach is that if a message pre-

[2] N-gram approach is often unbeatable (and therefore great) in many text classification tasks.

[3] "feature" indicates n-grams in our corpus that we use in the SVM classification step.

[4] The sentence breaker that we used is available at http://search.cpan.org/ ~shlomoy/Lingua-EN-sentence -0.25/lib/Lingua/EN/Sentence.pm.

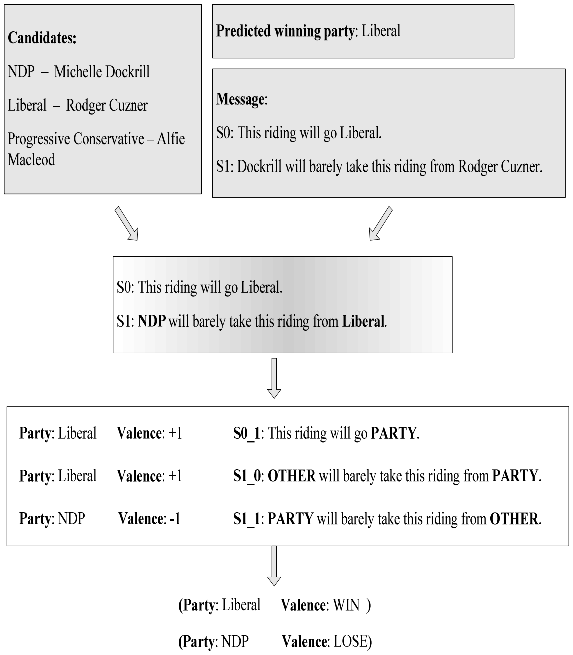

Candidates:
NDP – Michelle Dockrill
Liberal – Rodger Cuzner
Progressive Conservative – Alfie Macleod
Predicted winning party: Liberal
Message:
S0: This riding will go Liberal.
S1: Dockrill will barely take this riding from Rodger Cuzner.
S0: This riding will go Liberal.
S1: NDP will barely take this riding from Liberal.
Party: Liberal Valence: +1 S0_1: This riding will go PARTY.
Party: Liberal Valence: +1 S1_0: OTHER will barely take this riding from PARTY.
Party: NDP Valence: -1 S1_1: PARTY will barely take this riding from OTHER.
(Party: Liberal Valence: WIN)
(Party: NDP Valence: LOSE)

Figure 2. An example of feature generalization of a message

dicts a particular party to win, sentences which mention that party in the message also imply that it will win. Conversely all other parties are assumed to be in sentences that imply they will lose. As shown in Section 3.2, a message (*M*) in our corpus has a label of a party (P_w) that the author of *M* predicts to win. After breaking sentences in *M*, we duplicate a sentence by the number of unique parties in the sentence and modify the duplicated sentences by substituting the party names with PARTY and OTHER in order to generalize features.

Consider the following sentence:

"**Dockrill** will barely take this riding from **Rodger Cuzner**"
which gets re-written as:

"**NDP** will barely take this riding from **Liberal**"
because Dockrill is an NDP candidate and Rodger Cuzner is a Liberal candidate. Since the sentence contains two parties (i.e., NDP and Liberal), the algorithm duplicates the sentence twice, once for each party (see Lines 4–8 in Table 3)[5]. For NDP, the algorithm determines its Valence as -1 because NDP is not equal to the predicted winning party (i.e., Liberal) of the message (see Lines 4–5 in Table 3). Then it generates a generalized sentence by substituting NDP with PARTY and Liberal with OTHER (Lines 6–7). It returns (NDP, -1, "PARTY will barely take this riding from OTHER"). For Liberal, on the other hand, the algorithm determines its Valence as +1 since Liberal is the same as the predicted winning party of the message. After similar generalization, it returns (Liberal, +1, "OTHER will barely take this riding from PARTY").

Note that the final result of the feature generalization algorithm is a set of triplets: (Party, Valence, Generalized Sentence). Among a triplet, we use (Valence, Generalized Sentence) to produce feature vectors for a machine learning algorithm (see Section 4.2) and (Party, Valence) to integrate system results of each sentence for the final decision of Party and Valence of a message (see Section 4.3). Figure 2 shows an example of the algorithm.

4.2 Classification Using SVMs

In this step, we use Support Vector Machines (SVMs) to train our system using the generalized features described in Section 4.1. After we obtained examples of (Valence, Generalized Sentence) in the feature generalization step, we modeled a subtask of classifying a Generalized Sentence into Valence towards our final goal of determining (Valence, Party) of a message. This subtask is a binary classification since Valence has only 2 classes: +1 and -1[6]. Given a generalized sentence "OTHER will barely take this riding from PARTY" in Figure 2, for example, the goal of our system is to learn *WIN* valence for *PARTY*. Features for SVMs are extracted from generalized sentences. We implemented our SVM learning model using the SVM^{light} package[7].

4.3 SVM Result Integration

In this step, we combine the valence of each sentence predicted by SVMs to determine the final valence and predicted party of a message. For each party mentioned in a message, we calculate the sum of the party's valences of each sentence and

[5] In the feature generalization algorithm, we represent WIN and LOSE valence as +1 and -1.

[6] However, the final evaluation of the system and all the baselines is equally performed on the multi-classification results of messages.

[7] SVM^{light} is available from http://svmlight.joachims.org/

pick a party that has the maximum value. This integration algorithm can be represented as follows:

$$\arg\max_{p} \sum_{k=0}^{m} Valence_{k}(p)$$

where p is one of parties mentioned in a message, m is the number of sentences that contains party p in a message, and $Valence_k(p)$ is the valence of p in the kth sentence that contains p. Given the example in Figure 2, the *Liberal* party appears twice in sentence S0 and S1 and its total valence score is +2, whereas the *NDP* party appears once in sentence S1 and its valence sum is -1. As a result, our algorithm picks liberal as the winning party that the message predicts.

5 Experiments and Results

This section reports our experimental results showing empirical evidence that *Crystal* outperforms several baseline systems.

5.1 Experimental Setup

Our corpus consists of 4858 and 4680 messages from 2004 and 2006 Canadian federal election prediction data respectively described in detail in Section 3.2. We split our pre-processed corpus into 10 folds for cross-validation. We implemented the following five systems to compare with *Crystal* [8].

- **NGR**: In this algorithm, we train the system using SVM with n-gram features without the generalization step described in Section 4.1[9]. The replacement of each candidate's first and last name by his or her party name was still applied.
- **FRQ**: This system picks the most frequently mentioned party in a message as the predicted winning party. Party name substitution is also applied. For example, given a message "This riding will go liberal. Dockrill will barely take this riding from Rodger Cuzner.", all candidates' names are replaced by party names (i.e., "This riding will go Liberal. NDP will barely take this riding from Liberal."). After name replacement, the system picks Liberal as an answer because Liberal appears twice whereas NDP appears only once. Note that, unlike *Crystal*, this system does not consider the valence of each party (as done in our sentence duplication step of the feature generalization algorithm). Instead, it blindly picks the party that appeared most in a message.
- **MJR**: This system marks all messages with the most dominant predicted party in the entire data set. In our corpus, Conservatives was the majority party (3480 messages) followed closely by Liberal (3473 messages).
- **INC**: This system chooses the incumbent party as the predicted winning party of a message. (This is a strong baseline since incumbents often win in Canadian politics). For example, since the incumbent party of the riding "Blackstrap" in 2004 was Conservative, all the messages about Blackstrap in 2004 were marked Conservative as their predicted winning party by this system.
- **JDG**: This system uses judgment opinion words as its features for SVM. For our list of judgment opinion words, we use *General Inquirer* which is a publicly available list of 1635 positive and negative sentiment words (e.g., love, hate, wise, dumb, etc.)[10].

5.2 Experimental Results

We measure the system performance with its accuracy in two different ways: accuracy per message ($Acc_{message}$) and accuracy per riding (Acc_{riding}). Both accuracies are represented as follows:

$$Acc_{message} = \frac{\text{\# of messages the system correctly labled}}{\text{Total \# of messages in a test set}}$$

$$Acc_{riding} = \frac{\text{\# of ridings the system correctly predicted}}{\text{Total \# of ridings in a test set}}$$

We first report the results with $Acc_{message}$ in Evaluation1 and then report with Acc_{riding} in Evaluation2.

Evaluation1: Table 4 shows accuracies of baselines and *Crystal*. We calculated accuracy for each test set in 10-fold data sets and averaged it. Among the baselines, MJR performed worst (36.48%). Both FRQ and INC performed around 50% (54.82% and 53.29% respectively). NGR achieved its best score (62.02%) when using unigram, bigram, and trigram features together (uni+bi+tri). We also experimented with other feature combinations (see Table 5). Our system achieved 73.07% which is 11% higher than NGR and around 20%

[8] In our experiments using SVM, we used the linear kernel for all *Crystal*, NGR, and JDG.

[9] This system is exactly like *Crystal* without the feature generalization and result integration steps.

[10] Available at http://www.wjh.harvard.edu/~inquirer /homecat.htm

system	$\mathbf{Acc_{message}}$ (%)	$\mathbf{Acc_{riding}}$ (%)
FRQ	54.82	63.14
MJR	36.48	36.63
INC	53.29	78.03
NGR (uni+bi+tri)	62.02	79.65
JDG	66.23	78.68
Crystal (uni+bi+tri)	**73.07**	**81.68**

Table 4. System performance with accuracy per message ($Acc_{message}$) and accuracy per riding (Acc_{riding}): FRQ, MJR, INC, NGR, JDG, and *Crystal*.

Features	$\mathbf{Acc_{message}}$ (%)	
	NGR	***Crystal***
uni	60.49	72.03
bi	58.79	71.81
tri	54.04	69.57
four	47.25	67.64
uni + bi	61.54	72.93
uni + tri	61.36	72.20
uni + four	60.70	72.84
bi + tri	58.68	72.26
bi + four	58.54	72.17
uni + bi + tri	**62.02**	**73.07**
uni + bi + four	61.75	72.30
uni + tri + four	61.34	72.30
bi + tri + four	58.42	72.62
uni + bi + tri + four	61.96	73.01

Table 5. System performance with different features: Pure n-gram (NGR) and Generalized n-gram *Crystal*.

higher than FRQ and INC. The best accuracy of our system was also obtained with the combination of unigram, bigram, and trigram features.

The JDG system, which uses positive and negative sentiment word features, had 66.23% accuracy. This is about 7% lower than *Crystal*. Since the lower performance of JDG might be related to the number of features it uses, we also experimented with the reduced number of features of *Crystal* based on the *tfidf* scores[11]. With the same number of features (i.e., 1635), *Crystal* performed 70.62% which is 4.4% higher than JDG. An interesting finding was that NGR with 1635 features performed only 54.60% which is significantly lower than both systems. This indicates that the 1635 pure n-gram features are not as good as the same number of sentiment words carefully chosen from a dictionary but the generalized features of *Crystal* represent the predictive opinions better than JDG features.

Patterns in *WIN* class	**Patterns in *LOSE* class**
PARTY_will_win	want_OTHER
PARTY_hold	PARTY_don't_have
PARTY_will_win_this	OTHER_and
PARTY_win	the_PARTY
will_go_PARTY	OTHER_will_win
PARTY_will_take	OTHER_is
PARTY_will_take_this	to_the_OTHER
PARTY_is	and_OTHER
safest_PARTY	results_OTHER
PARTY_has	OTHER_has
go_PARTY_again	to_OTHER

Table 6. Examples of frequent features in WIN and LOSE classes.

Table 5 illustrates the comparison of NGR (without feature generalization) and *Crystal* (with feature generalization) in different feature combinations. *uni, bi, tri,* and *four* correspond to *unigram, bigram, trigram*, and *fourgram*. Our proposed technique *Crystal* performed always better than the pure n-gram system (NGR). Both systems performed best (62.02% and 73.07%) with the combination of unigram, bigram, and trigram (uni+bi+tri). The second best scores (61.96% and 73.01%) are achieved with the combinations of all grams (uni+bi+tri+four) in both systems. Using fourgrams alone performed worst since the system overfitted to the training examples.

Table 6 presents several examples of frequent n-gram features in both WIN and LOSE classes. As shown in Table 6, lexical patterns in the WIN class express optimistic sentiments about PARTY (e.g., PARTY_will_win and go_ PARTY_again) whereas patterns in the LOSE class express pessimistic sentiments (e.g., PARTY_don't_have) and optimistic ones about OTHER (e.g., want_OTHER).

Evaluation2: In this evaluation, we use Acc_{riding} computed as the number of ridings that a system correctly predicted, divided by the total number of ridings. For each riding *R*, systems pick a party that obtains the majority prediction votes from messages in *R* as the winning party of *R*. For ex-

[11] The total number of all features of *Crystal* is 689,642.

ample, if *Crystal* identified 9 messages predicting for Conservative Party, 3 messages for NDP, and 1 message for Liberal among 13 messages in the riding "Blackstrap", the system will predict that the Conservative Party would win in "Blackstrap".

Table 4 shows the system performance with Acc_{riding}. Note that people who write messages on a particular web site are not a random sample for prediction. So we introduce a measure of confidence (*ConfidenceScore*) of each system and use the prediction results when the *ConfidenceScore* is higher than a threshold. Otherwise, we use a default party (i.e., the incumbent party) as the winning party. *ConfidenceScore* of a riding *R* is calculated as follows:

$$ConfidenceScore = count_{message}(P_{first}) - count_{message}(P_{second})$$

where $count_{message}(P_x)$ is the number of messages that predict a party P_x to win, P_{first} is the party that the most number of messages predict to win, and P_{second} is the party that the second most number of messages predict to win.

We used 62 ridings to tune the *ConfidenceScore* parameter arriving at the value of 4. As shown in Table 4, the system which just considers the incumbent party (INC) performed fairly well (78.03% accuracy) because incumbents are often re-elected in Canadian elections. The upper bound of this prediction task is 88.85% accuracy which is the prediction result using numerical values of a prediction survey. FRQ and MJR performed 63.14% and 36.63% respectively. Similarly to Evaluation1, JDG which only uses judgment word features performed worse than both *Crystal* and NGR. Also, *Crystal* with our feature generalization algorithm performed better than NGR with non-generalized n-gram features. The accuracy of *Crystal* (81.68%) is comparable to the upper bound 88.85%.

6 Discussion

In this section, we discuss possible extensions and improvements of this work.

Our experiment focuses on investigating aspects of predictive opinions by learning lexical patterns and comparing them with judgment opinions. However, this work can be extended to investigating how those two types of opinions are related to each other and whether lexical features of one (e.g., judgment opinion) can help identify the other (e.g., predictive opinion). Combining two types of opinion features and testing on each domain can examine this issue.

In our experiment, we used General Inquirer words as judgment opinion indicators for JDG baseline system. It might be interesting to employ different resources for judgment words such as the polarity lexicon by Wilson et al. (2005) and the recently released SentiWordNet[12].

Our work is an initial step towards analyzing a new type of opinion. In the future, we plan to incorporate more features such as priors like incumbent party in addition to the lexical features to improve the system performance.

7 Conclusions

In this paper, we proposed a framework for working with *predictive opinion*. Previously, researchers in opinion analysis mostly focused on judgment opinions which express positive or negative sentiment about a topic, as in product reviews and policy discussions. Unlike judgment opinions, predictive opinions express a person's opinion about the future of a topic or event such as the housing market, a popular sports match, and election results, based on his or her belief and knowledge. Among these many kinds of predictive opinions, we focused on election prediction.

We collected past election prediction data from an election prediction project site and automatically built a gold standard. Using this data, we modeled the election prediction task using a supervised learning approach, SVM. We proposed a novel technique which generalized n-gram feature patterns. Experimental results showed that this approach outperforms several baselines as well as a non-generalized n-gram approach. This is significant because an n-gram model without generalization is often extremely competitive in many text classification tasks.

This work adopts NLP techniques for predictive opinions and it sets the foundation for exploring a whole new subclass of the opinion analysis problems. Potential applications of this work are systems that analyze various kinds of election predictions by monitoring texts in discussion boards and personal blogs. In the future, we would like to

[12] http://sentiwordnet.isti.cnr.it/

model predictive opinions in other domains such as the real estate market and the stock market which would require further exploration of system design and data collection.

Reference

Engelmore, R., and Morgan, A. eds. 1986. *Blackboard Systems*. Reading, Mass.: Addison-Wesley.

Dave, K., Lawrence, S. and Pennock, D. M. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. *Proc. of World Wide Web Conference 2003*

Drucker, H., Wu, D. and Vapnik, V. 1999. Support vector machines for spam categorization. *IEEE Trans.* Neural Netw., 10, pp 1048–1054.

Gliozzo, A., Strapparava C. and Dagan, I. 2005. Investigating Unsupervised Learning for Text Categorization Bootstrapping, *Proc. of EMNLP 2005*. Vancouver, B.C., Canada

Hu, M. and Liu, B. 2004. Mining and summarizing customer reviews. *Proc. Of KDD-2004*, Seattle, Washington, USA.

Jindal, N. and Liu, B. 2006. Mining Comprative Sentences and Relations. *Proc. of 21st National Conference on Artificial Intellgience (AAAI-2006).* 2006. Boston, Massachusetts, USA

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features, *Proc. of ECML*, p. 137–142.

Kim, S-M. and Hovy, E. 2004. Determining the Sentiment of Opinions. *Proc. of COLING 2004*.

Liu, B., Li, X., Lee, W. S. and Yu, P. S. Text Classification by Labeling Words *Proc. of AAAI-2004*, San Jose, USA.

Pang, B, Lee, L. and Vaithyanathan, S. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proc. of EMNLP 2002*.

Popescu, A-M. and Etzioni, O. 2005. Extracting Product Features and Opinions from Reviews, *Proc. of HLT-EMNLP 2005*.

Rickel, J. and Porter, B. 1997. Automated Modeling of Complex Systems to Answer Prediction Questions, *Artificial Intelligence Journal*, volume 93, numbers 1-2, pp. 201–260

Riloff, E., Wiebe, J., and Phillips, W. 2005. Exploiting Subjectivity Classification to Improve Information Extraction, *Proc. of the 20th National Conference on Artificial Intelligence (AAAI-05)* .

Riloff, E., Wiebe, J. and Wilson, T. 2003. Learning Subjective Nouns Using Extraction Pattern Bootstrapping. *Proc. of CoNLL 2003*. pp 25–32.

Rodionov, S. and Martin, J. H. 1996. A Knowledge-Based System for the Diagnosis and Prediction of Short-Term Climatic Changes in the North Atlantic, *Journal of Climate*, 9(8)

Turney, P. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *Proc. of ACL 2002*, pp 417–424.

Wiebe, J., Bruce, R. and O'Hara, T. 1999. Development and use of a gold standard data set for subjectivity classifications. *Proc. of ACL 1999*, pp 246–253.

Wiebe, J., Wilson, T. , Bruce, R. , Bell , M. and Martin, M. Learning Subjective Language. 2004. *Computational Linguistics*

Wilson, T., Wiebe, J. and Hoffmann, P. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. *Proc. of HLT/EMNLP 2005*.

Yu, H. and Hatzivassiloglou, V. 2003. Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. *Proc. of EMNLP 2003*.

Extracting Aspect-Evaluation and Aspect-of Relations in Opinion Mining

Nozomi Kobayashi [*] **Kentaro Inui, and Yuji Matsumoto**
Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192, Japan
{nozomi-k,inui,matsu}@is.naist.jp

Abstract

The technology of opinion extraction allows users to retrieve and analyze people's opinions scattered over Web documents. We define an opinion unit as a quadruple consisting of the opinion holder, the subject being evaluated, the part or the attribute in which the subject is evaluated, and the value of the evaluation that expresses a positive or negative assessment. We use this definition as the basis for our opinion extraction task. We focus on two important subtasks of opinion extraction: (a) extracting aspect-evaluation relations, and (b) extracting aspect-of relations, and we approach each task using methods which combine contextual and statistical clues. Our experiments on Japanese weblog posts show that the use of contextual clues improve the performance for both tasks.

1 Introduction

The explosive increase in Web communication has attracted increasing interest in technologies for automatically mining personal opinions from Web documents such as product reviews and weblogs. Such technologies would benefit users who seek reviews on certain consumer products of interest.

Previous approaches to the task of mining a large-scale document collection of customer opinions (or reviews) can be classified into two approaches: Document classification and information extraction. The former is the task of classifying documents or passages according to their semantic orientation such as positive vs. negative. This direction has been forming the mainstream of research on opinion-sensitive text processing (Pang et al., 2002; Turney, 2002, etc.). The latter, on the other hand, focuses on the task of extracting opinions consisting of information about, for example, ⟨*who* feels *how* about *which aspect* of *what product*⟩ from unstructured text data. In this paper, we refer to this information extraction-oriented task as *opinion extraction*. In contrast to sentiment classification, opinion extraction aims at producing richer information and requires an in-depth analysis of opinions, which has only recently been attempted by a growing but still relatively small research community (Yi et al., 2003; Hu and Liu, 2004; Popescu and Etzioni, 2005, etc.).

Most previous work on customer opinion extraction assumes the source of information to be customer reviews collected from customer review sites (Popescu and Etzioni, 2005; Hu and Liu, 2004; Liu et al., 2005). In contrast, in this paper, we consider the task of extracting customer opinions from unstructured weblog posts. Compared with extraction from review articles, extraction from weblogs is more challenging because weblog posts tend to exhibit greater diversity in topics, goals, vocabulary, style, etc. and are much more likely to include descriptions irrelevant to the subject in question. In this paper, we first describe our task setting of opinion extraction. We conducted a corpus study and investigated the feasibility of the task def-

[*] Currently, NTT Cyber Space Laboratories,
1-1, Hikarinooka, Yokosuka, Kanagawa, 239-0847 Japan

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1065–1074, Prague, June 2007.

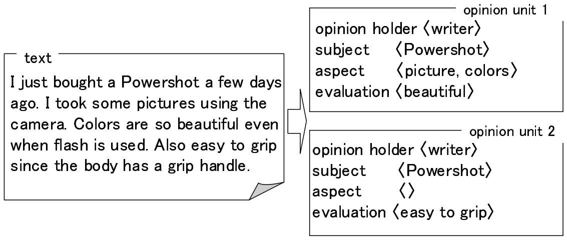
inition by showing the statistics and inter-annotator agreement of our corpus annotation. Next, we show that the crucial body of the above opinion extraction task can be decomposed into two kinds of relation extraction, i.e. aspect-evaluation relation extraction and aspect-of relation extraction. For example, the passage "*I went out for lunch at the Deli and ordered a curry with chicken. It was pretty good*" has an aspect-evaluation relation ⟨*curry with chicken, was good*⟩ and an aspect-of relation ⟨*The Deli, curry with the chicken*⟩. The former task can be regarded as a special type of predicate-argument structure analysis or semantic role labeling. The latter, on the other hand, can be regarded as bridging reference resolution (Clark, 1977), which is the task of identifying relations between definite noun phrases and discourse-new entities implicitly related to some previously mentioned entities.

Most of the previous work on customer opinion extraction, however, does not adopt the state-of-the-art techniques in those fields, relying only on simple proximity-based or pattern-based methods. In this context, this paper empirically shows that incorporating machine learning-based techniques devised for predicate-argument structure analysis and bridging reference resolution improve the performance of both aspect-evaluation and aspect-of relation extraction. Furthermore, we also show that combining contextual clues with a common co-occurrence statistics-based technique for bridging reference resolution makes a significant improvement on aspect-of relation extraction.

2 Opinion extraction: Task design

Our present goal is to build a computational model to extract opinions from Web documents in such a form as: *Who* feels *how* on *which aspects* of *which subjects*. Given the passage presented in Figure 1, for example, the opinion we want to extract is: "*the writer* feels that *the colors* of *pictures* taken with *Powershot* (product) are *beautiful*." As suggested by this example, we consider it reasonable to start with an assumption that most evaluative opinions can be structured as a frame composed of the following constituents:

Opinion holder The person who is making an evaluation. An opinion holder is typically the first person (the author). We say the opinion holder is unspecified if the opinion is mentioned as a rumor.



text

I just bought a Powershot a few days ago. I took some pictures using the camera. Colors are so beautiful even when flash is used. Also easy to grip since the body has a grip handle.

opinion unit 1

opinion holder ⟨writer⟩
subject ⟨Powershot⟩
aspect ⟨picture, colors⟩
evaluation ⟨beautiful⟩

opinion unit 2

opinion holder ⟨writer⟩
subject ⟨Powershot⟩
aspect ⟨⟩
evaluation ⟨easy to grip⟩

Figure 1: Extraction of opinion units

Subject A named entity (product or company) of a given particular class of interest (e.g. a car model name in the automobile domain).

Aspect A part, member or related object, or an attribute (of a part) of the subject on which the evaluation is made (*engine*, *size*, etc.)

Evaluation An evaluative or subjective phrase used to express an evaluation or the opinion holder's mental/emotional attitude (*good, poor, powerful, stylish, (I) like, (I) am satisfied*, etc.)

According to this typology, the example in Figure 1 has six constituents, *the writer* (opinion holder), *Powershot* (subject), *pictures* (aspect), *colors* (aspect), *beautiful* (evaluation), *easy to grip* (evaluation), and constitute two units of opinions as presented in the right half of the figure. We call such a unit an *opinion unit*. In this paper, we only consider explicitly mentioned evaluative opinions as our targets of extraction, excluding opinions indirectly expressed through, for example, style or language choice from our scope.

Under this assumption, opinion extraction can be defined as a task of filling a fixed number of slots as above for each of the evaluations expressed in a given text collection. Two issues then immediately arise. First, it is necessary to make sure that the definition of the opinion units is clear enough for human annotators to be able to carry out the task with sufficient accuracy. Second, all the slots might not consist of simple expressions in that the filler of an aspect slot may have a hierarchical structure in itself. For example, "*the leather cover of the seats (of a car)*" refers to a part of a part of a car. In theory, such a hierarchical chain can be of any length, which may affect the feasibility of the task. For tackling these issues, we built a corpus annotated with the above sort of information and investigated the feasibility of the task.

2.1 Corpus study

We first collected 116 Japanese weblog posts in the restaurant domain by randomly sampling from a collection of posts classified under the "gourmet" category on a major blog site: http://blog.livedoor.com/.

We asked two annotators to annotate them independently of each other following the above specification. The annotators first identified evaluative phrases, and then for each evaluative phrase judged whether it was concerning a particular subject (i.e. a restaurant) in the given domain. If judged yes, the annotators filled the opinion holder and subject slots obligatorily. The annotators filled the aspect slot only when its filler appeared in the document and identified the hierarchical relations between aspects if any (e.g. *noodle* and its *volume*). Note that, if a sentence has two or more evaluations, they have to make one opinion unit for each.

2.1.1 Inter-annotator agreement

We investigated the degree of inter-annotator agreement. In the task of identifying evaluations, one annotator A_1 identified 450 evaluations while the other A_2 identified 392, and 329 cases of them coincided. The two annotators did not identify the same number of evaluations, so instead of using $kappa$ statistics, we use the following metric for measuring agreement as Wiebe et al. (2005) do:

$$agr(A_1||A_2) = \frac{\text{\# of tags agreed by } A_1 \text{ and } A_2}{\text{\# of tags annotated by } A_1}$$

$agr(A_1||A_2)$ was 0.73 and $agr(A_2||A_1)$ was 0.83. The F1 measure of the agreement between the two was therefore 0.79, which indicate that humans can identify evaluation at a reasonable level.

Next, we investigated the inter-annotator agreement of the aspect-evaluation and subject-evaluation relations. Annotator A_1 identified 328 relations, and A_2 identified 346 relations. 295 cases coincided, and $agr(A_2||A_1)$ was 0.90 and $agr(A_1||A_2)$ was 0.86 (F1 measure was 0.88). This shows that we obtained high consistency. Finally, for the subject-aspect and aspect-aspect relations, annotator A_1 identified 296 relations, while A_2 identified 293, 233 cases of which got agreement. $agr(A_2||A_1)$ was 0.79 and $agr(A_1||A_2)$ was 0.80 (F1 measure was 0.79), which show that the human annotators can carry out the task at a reasonable accuracy. Based on this corpus study, we believe that our definitions of two relations are clear enough for constructing annotated corpus.

Table 1: Statistics of opinion-annotated corpus (Restaurant, Automobile, cellular phone and video game)

		Rest	Auto	Phone	Game
	articles	1,356	564	481	361
	sentences	21,666	14,005	11,638	6,448
	# of opinion units	4,267	1,519	1,518	775
I	Asp-Eval	3,692	943	965	521
	Asp-Asp	1,426	280	296	221
	Subj-Asp	2,632	877	850	451
II	Subj-Eval	575	576	553	243
	Subj-Asp-Eval	2,314	736	768	351
	Subj-Asp-Asp-Eval	1,065	175	172	127
	other	313	32	25	54
Non-writer op. holder		95	17	22	2

2.1.2 Opinion-annotated corpus

Based on these results, we collected a larger set of weblog posts in four domains: restaurant, automobile, cellular phone and video game. We then asked annotator A_1 to annotate them in the same annotation scheme as above. The results are summarized in Table 1. I in the table shows the number of the identified opinion units and relations, and II shows the number of hierarchical chains of aspects. For example, "*Nokia 6800 has a nice color screen*" is counted as "Subj-Asp-Eval" since this example includes a subject "Nokia 6800", an aspect "color screen" and an evaluation "nice". "Other" indicates the number of the case where the length of hierarchical chains of aspects is three or more. One observation is that, for all the domains, 90 % of all the opinion units have a hierarchical chain of aspects whose length is two or less. From this, we can conclude that hierarchical chains longer than two are rare, and the problem is not so complicated, though they can be of any length in theory.

The row of "Non-writer op(inion) holder" at the bottom of Table 1 shows the number of opinion units whose opinion holder is *not* the writer of the weblog. This result indicates that when an evaluative expression is found, its opinion holder is highly likely to be the writer of the blogs. Therefore, we put aside the task of filling the opinion holder slot in this paper.

2.2 Related work on task settings of opinion extraction

There are several researches on customer opinion extraction. Hu and Liu (2004) considered the task of extracting ⟨*Aspect, Sentence, Semantic-orientation*⟩ triples in our terminology, where *Sentence* is the one that includes the *Aspect*, and *Semantic-orientation* is either positive or negative.

The notion of Evaluation in our term has also been introduced by previous work (Popescu and Etzioni, 2005; Tateishi et al., 2004; Suzuki et al., 2006; Kobayashi et al., 2005, etc.). For example, our previous paper (Kobayashi et al., 2005) addresses the task of extracting ⟨Subject,Aspect,Evaluation⟩. However, none of those papers reports on such an extensive corpus study as what we report in this paper. In addition, in this paper, we consider not only aspect-evaluation relations but also hierarchical chains of subject-aspect and aspect-aspect relations, which has never been addressed in previous work.

Open-domain opinion extraction is another trend of research on opinion extraction, which aims to extract a wider range of opinions from such texts as newspaper articles (Yu and Hatzivassiloglou, 2003; Kim and Hovy, 2004; Wiebe et al., 2005; Choi et al., 2006). To the best of our knowledge, one of the most extensive corpus studies in this field has been conducted in the MPQA project (Wiebe et al., 2005); while their concerns include the types of opinions we consider, they annotate newspaper articles, which presumably exhibit considerably different characteristics from customer-generated texts.

Though we do not discuss the problem of determining semantic orientation, we assume availability of state-of-the-art methods that perform this task (Suzuki et al., 2006; Takamura et al., 2006, etc.). The problem of determining semantic orientation will be solved by using these techniques, so we focus on the main issue: Extracting opinion units from given texts.

3 Method for opinion extraction

Before designing a model for our opinion extraction task, it is important to note that aspect phrases are open-class expressions and tend to be heavily domain-dependent. In fact, according to our investigation on our opinion-annotated corpus, the number of aspect types is nearly 3,200, and only 3% of them appear in two or more domains as shown in Figure 2. For evaluation expressions, on the other hand, the number of types is much smaller than that of aspect expressions, and 27% of them appear in multiple domains. This indicates that evaluation expressions are more likely to be used commonly across different domains compared with aspects.

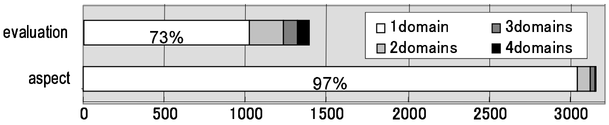


Figure 2: The distributions of evaluation and aspect expressions in the four domains

To prove this assumption, we created a dictionary of evaluation expressions from customer reviews of automobiles (230,000 sentences in total) using the semi-automatic method proposed by Kobayashi et al. (2004). We expanded the dictionary by hand with external resources including publicly available ordinal thesauri. As a result, we collected 5,550 entries. According to our investigation of the coverage by the dictionary, 0.84 (restaurant), 0.88 (cellular phone), 0.91 (automobile), and 0.93 (video game) of the evaluations annotated in our corpus are covered by the dictionary. From this observation, we consider that it is reasonable to start opinion extraction with the identification of evaluation expressions. We therefore design the process of extracting ⟨Subject, Aspect, Evaluation⟩ as follows:

1. **Aspect-evaluation relation extraction**: For each of the candidate evaluations that are selected from a given document by dictionary look-up, identify the target of the evaluation. Here the identified target may be a subject (e.g. *IXY (is well-designed)*) or an aspect of a subject (e.g. *the quality (is amazing)*). Hereafter, we use the term *aspect* to refer to both an aspect and a subject itself, since the subject can be regarded as the top element in the hierarchical chain of aspects.
2. **Opinion-hood determination**: Judge whether or not the obtained pair ⟨aspect, evaluation⟩ is an expression of an opinion by considering the given context. If it is judged yes, go to step3; otherwise, return to step 1 with a new candidate

evaluation expression.

3. **Aspect-of relation extraction**: If the identified aspect is not a subject, search for its antecedent, i.e. an expression that is a higher aspect or a subject of the current aspect. Repeat step 3 until reaching a subject or no parent is found.

3.1 Related work on opinion extraction

A common approach to the customer opinion extraction task mainly uses simple proximity- or pattern-based techniques. For example, Tateishi et al. (2004) implement five syntactic patterns and Popescu et al. (2005) use ten syntactic patterns. Such an approach is limited in two respects. First, it assumes the availability of a list of potential aspect expressions as well as evaluation expressions; however creating such a list of aspects for a variety of domains can be prohibitively expensive because of the domain dependency of aspect expressions. In contrast, our method does not require any aspect lexicon.

Second, their approach lacks the perspective of viewing aspect-evaluation extraction as a specific type of predicate-argument structure analysis, i.e. the task of identifying the arguments of a given predicate in a given text, and fails to benefit from the state-of-the-art techniques of this rapidly growing field. The syntactic patterns used in their research are analyzed by a dependency parser, however, aspect-evaluation relations appear in diverse syntactic patterns, which cannot be easily captured by a handful of manually devised rules.

An exception is the model reported by Kanayama et al.(2004), which uses a component of an existing MT system to identify the "aspect" argument of a given "evaluation" predicate. However, the MT component they use is not publicly available, and even if it were, it would be difficult to apply it to tasks in hand due of the opaqueness of its mechanism. Our approach aims to develop a more generally applicable model of aspect-evaluation extraction.

In open-domain opinion extraction, some approaches use syntactic features obtained from parsed input sentences (Choi et al., 2006; Kim and Hovy, 2006), as is commonly done in semantic role labeling. Choi et al. (2006) address the task of extracting opinion entities and their relations, and incorporate syntactic features to their relation extraction model. Kim and Hovy (2006) proposed a method for extracting opinion holders, topics and opinion words, in which they use semantic role labeling as an intermediate step to label opinion holders and topics. However, these approaches do not address the task of extracting aspect-of relations and make use of syntactic features only for labeling opinion holders and topics. In contrast, as we describe below, we find the significant overlap between aspect-evaluation relation extraction and aspect-of relation extraction and apply the same approach to both tasks, gaining the generality of the model.

Aspect-of relations can be regarded as a sub-type of bridging reference (Clark, 1977), which is a common linguistic phenomenon where the referent of a definite noun phrase refers to a discourse-new entity implicitly related to some previously mentioned entity. For example, we can see a relation of bridging reference between "*the door*" and "*the room*" in "*She entered the room. The door closed automatically.*" A common approach is to use co-occurrence statistics between the referring expression (e.g. "*the door*" in the above example) and the related entity ("*the room*") (Bunescu, 2003; Poesio et al., 2004). Our approach newly incorporates automatically induced syntactic patterns as contextual clues into such a co-occurrence model, producing significant improvements of accuracy.

3.2 Our approach

Now we describe our approach to aspect-evaluation and aspect-of relation extraction. The key idea is to combine the following two kinds of information using a machine learning technique for both tasks.

Contextual clues: Syntactic patterns such as

⟨Aspect⟩-ga *VP-te,* *⟨Evaluation⟩*
⟨Aspect⟩-NOM *VP*-CONJ *⟨Evaluation⟩*

which matches such a sentence as

⟨sekkyaku⟩-ga kunrens-aretei-te ⟨kimochiyoi⟩
⟨service⟩-NOM *be trained*-CONJ *⟨feel comfortable⟩*
(*The waiters were well-trained, so I felt comfortable.*)

are considered to be useful for extracting relations between slot fillers when they appear in a single sentence (Here, ⟨⟩ indicates a slot filler). We employ a supervised learning technique to search for such useful syntactic patterns.

Context-independent statistical clues: Statistics such as aspect-aspect and aspect-evaluation

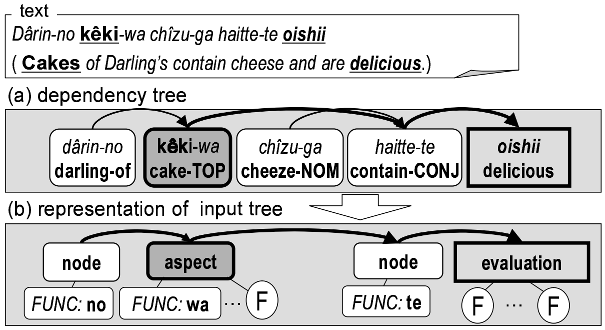


Figure 3: Representation of input data

co-occurrences are expected to be useful. We obtain such statistical clues automatically from a large collection of raw documents.

In what follows, we describe our method for aspect-evaluation. The aspect-of relation extraction is done in an an analogous way.

3.2.1 Supervised learning of contextual clues

Let us consider the problem of searching for the aspect of a given evaluation expression t. This problem can be decomposed into binary classification problems of deciding whether each pair of candidate aspect c and target t is in an aspect-evaluation relation or not. Our goal is to learn a discrimination function for this classification problem. If such a function is obtained, we can identify the most likely candidate aspect simply by selecting the best scored c-t pair and, if its score is negative for all possible candidates, we conclude that t has no corresponding aspect in the candidate set.

For finding syntactic patterns that extract an aspect c starting with an evaluation t, we first represent all the sentences in the annotated corpus that has both an aspect and its evaluation, as shown in Figure 3. A sentence is analyzed by a dependency parser, then the dependency tree is converted so as to represent the relation between content words clearly and to attach other information (such as POS labels and other morphological features of content words and the functional words attached to the content words) as shown in the lower part of Figure 3. Among various classifier induction algorithms for tree-structured data, in our experiments, we have so far examined Kudo and Matsumoto (2004)'s algorithm, packaged as a free software named *BACT*. Given a set of training examples represented as ordered trees labeled either positive or negative class, this algorithm learns a list of weighted decision stumps as a discrimination function with the Boosting algorithm. Each decision stump is associated with tuple $\langle s, l, w \rangle$, where s is a subtree appearing in the training set, l a label, and w a weight of this pattern. The strength of this algorithm is that it automatically acquires structured features and allows us to analyze the utility of features.

Given a c-t pair in an annotated sentence, tree encoding of this sentence is done as follows: First, we use a dependency parser to obtain a dependency tree as in Figure 3 (a). We assume “*kêki* (*cake*)” as the candidate aspect c and “*oishii* (*delicious*)” as the target evaluation t. We then find the path between t and c together with their daughter nodes. For example, the node “*Darling-no* (*Darling's*)” is kept since it is a daughter of c. Then, all the content words are abstracted to either of the class types, evaluation, aspect or node, that is, c is renamed as ”aspect”, t as ”evaluation” and all other content words as ”node”. Other information of a content word and the information of functional words attaching to the content word are represented as the leaf nodes as shown in Figure 3 (b). The features used in our experiments are summarized in Table 2.

We apply the same method to the aspect-of relation extraction by replacing the ”evaluation” label as the second ”aspect” label.

3.3 Context-independent statistical clues

We also introduce the following two kinds of statistical clues.

i. Co-occurrences of aspect-evaluation/aspect-aspect: Among various ways to estimate the strength of association (e.g. the number of hits returned from a search engine), in our experiments, we extracted aspect-aspect and aspect-evaluation co-occurrences in 1.7 million weblog posts using the patterns “⟨aspect⟩ *ga/wa/mo* ⟨evaluation⟩ (⟨aspect⟩ *is (subject-marker)* ⟨evaluation⟩)” and “⟨aspect_A⟩ *no* ⟨aspect_B⟩ ga/wa (⟨aspect_B⟩ *of* ⟨aspect_A⟩ *is*)”. To avoid the data sparseness problem, we use Probabilistic Latent Semantic Indexing (PLSI) (Hofmann, 1999) to estimate conditional probabilities $P(Aspect|Evaluation)$ and $P(Aspect\_A|Aspect\_B)$. We then incorporate the information of these probability scores into the learning model described in 3.2 by encoding them as a feature that indicates the relative score rank of each candidate in a given candidate set (see Table 2).

ii. Aspect-hood of candidate aspects: Aspect-hood is an index of the degree that measures how plausible a term is used as an aspect within a given domain. We consider that a phrase directly co-occurred with a subject often is likely to be an aspect of the subject, and extract the expression X which appears in the form "*Subject no X* (*X of Subject*)" and the expression Y which appears in the form "X *no* Y". We calculate the aspect-hood of the expressions X and Y by the pointwise mutual information. This score is also used as a features (see Table 2).

3.4 Intra-/inter-sentential relation extraction

Syntactic pattern induction as described in 3.2.1 can apply only when an aspect-evaluation (or aspect-of) relation appears in a single sentence. We therefore build a separate model for inter-sentential relation extraction, which is carried out after intra-sentential relation extraction.

1) Intra-sentential relation identification: Given a target evaluation (or aspect), select the most likely candidate aspect c^* within the target sentence with the intra-sentential model described in 3.2.1. If the score of c^* is positive, return c^*; otherwise, go to the inter-sentential relation extraction phase.

2) Inter-sentential relation identification: Search for the most likely candidate aspect in the sentences preceding the target evaluation (or aspect). This task can be regarded as a zero-anaphora resolution problem. For this purpose, we employ the supervised learning model for zero-anaphora resolution proposed by (Iida et al., 2003).

3.5 Opinion-hood determination

Evaluation phrases do not always extract correct opinion units in a given domain. Consider an example from the digital camera domain, "*The weather was good. so I went to the park to take some pictures*". "*good*" expresses the evaluation for "*the weather*", but "*the weather*" is not an aspect of digital cameras. Therefore, ⟨*the weather, good*⟩ is not an opinion in the digital camera domain. We can consider a binary classification task of judging whether the obtained opinion unit is a real opinion or not in a given domain. In this paper, we conduct a preliminary experiment which uses the opinion-hood determination model learned by Support Vector Machines. We conduct the model using our opinion-annotated corpus. The positive examples are aspect-evaluation pairs annotated in the corpus. The negative examples are artificially generated as follows: We first identify the expression in the evaluation dictionary that appear in our annotated corpus. We then apply the above aspect-evaluation extraction method and get the most plausible candidate aspect. The result is regarded as a negative example if the extracted aspect is not the true aspect. The features we used in our experiments are summarized in Table 2.

4 Experiments

We conducted experiments with our Japanese opinion-annotated corpus to empirically evaluate the performance of our approach. In these experiments, we separately evaluated the models of aspect-evaluation relation extraction, aspect-of relation extraction, and opinion-hood determination.

4.1 Common settings

We chose 395 weblog posts in the restaurant domain from our opinion-annotated corpus described in 2.1, and conducted 5-fold cross validation on that dataset. As preprocessing, we analyzed this corpus using the Japanese morphological analyzer *ChaSen*[1] and the Japanese dependency structure analyzer *CaboCha*[2].

4.2 Models

The results are summarized in Tables 3 and 4. We evaluated the results by recall R and precision P defined as follows

$$R = \frac{\text{correctly extracted relations}}{\text{total number of relations}},$$

$$P = \frac{\text{correctly extracted relations}}{\text{total number of relations found by the system}}.$$

Note that, in aspect-of relations, we permit ⟨A,C⟩ to be correct when the data includes the chain of aspect-of relations ⟨A,B⟩ and ⟨B,C⟩. Therefore, we merged the intra- and inter-sentential results as shown in Table 4.

[1] http://chasen.naist.jp/

[2] http://chasen.org/˜taku/software/cabocha/

Table 2: Feature list: t denotes a given target (evaluation or aspect) and c a candidate

Features for contextual clues

- Position of c / t in the sentence (beginning, end, other)
- Base phrase distance between c and t (1, 2, 3, 4, other)
- Whether c and t has a immediate dependency relation
- Whether c precedes t
- Whether c appears in a quoted sentence
- Part-of-speech of c / t
- Suffix of c (*-sei*, *-sa* (-ty), etc.)
- Character type of c (*English*, *Chinese*, *Katakana*, etc.)
- Semantic class of c derived from *Nihongo Goi Taikei* (Ikehara et al., 1997).

Features for statistical clues

- Co-occurrence score rank of c (1st, 2nd, 3rd, 4th, other)
- Aspect-hood score rank of c (1st, 2nd, 3rd, 4th, other)

Table 3: The results of aspect-evaluation relation

		intra-sent.	inter-sent.
Patterns	P	0.56 (432/774)	-
	R	0.53 (432/809)	-
Contextual	P	0.70 (504/723)	0.13 (46/360)
	R	0.62 (504/809)	0.17 (46/274)
Contextual +statistics	P	0.72 (502/694)	0.14 (53/389)
	R	0.62 (502/809)	0.19 (53/274)

Table 4: The results of aspect-of relation

	precision	recall
Co-occurrence	0.27 (175/ 682)	0.17 (175/1048)
Contextual	0.44 (458/1047)	0.44 (458/1048)
Contextual+statistics	0.45 (474/1047)	0.45 (474/1048)

The *Contextual* and *Contextual+statistics* models are our proposed models where the former uses only contextual clues (3.2.1) and the latter uses both contextual and statistical clues. We prepared two baseline models, one for each of the above tasks. The *Pattern* model (in Table 3) simulates the pattern-based method proposed by Tateishi at al. (2004), which uses the following patterns: "⟨Aspect⟩ case-particle ⟨Evaluation⟩" and "⟨Evaluation⟩ syntactically depends on ⟨Aspect⟩". The *Co-occurrence* model (in Table 4) simulates the co-occurrence statistics-based model used in bridging reference resolution (Bunescu, 2003): For an aspect expression, we select the nearest candidate that has the highest positive score of the pointwise mutual information regardless of its occurrence (i.e. inter- or intra-sentential). Comparing the *Pattern* (*Co-occurrence*) model with the *Contextual* model shows the effects of the supervised learning with contextual clues, while comparison of the *Contextual* and *Contextual+statistics* models shows the joint effect of combining contextual and statistical clues.

4.3 Results and discussions

As for the aspect-evaluation relation extraction, concerning the intra-sentential cases, we can see that the models using the contextual clues show nearly 10% improvement in both precision and recall. This indicates that the machine learning-based method has a great advantage over the pattern-based approach. Similar results are seen in aspect-of relation extraction. The models using the contextual clues achieved more than 10% improvement in precision and 20% improvement in recall over the co-occurrence statistics-based model. We can say that contextual clues are also useful in aspect-of relation extraction. In comparing the Contextual and Contextual+statistics models, on the other hand, we could get only a slight improvement, which indicates that we need to estimate the statistical clues more precisely. We found that the unsophisticated estimation of the statistical clues was a major source of errors in aspect-of relation extraction, however, this estimation is not so easy since the correct expressions are appeared only once in large data. We are seeking efficient ways to avoid data sparseness problem (e.g. categorize the aspects).

In the aspect-evaluation relation extraction, we evaluated the results against the human annotated gold-standard in a strict manner. However, according to our error analysis, some of the errors can be regarded as correct for some real applications. In the following example, a relation annotated by the human is "*aji (taste)*, *koi-me (strong)*".

misoshiru-wa ⟨*aji*⟩*-ga* ⟨*koi-me*⟩
miso soup-TOP ⟨*taste*⟩-NOM ⟨*strong*⟩
(*The taste of the miso soup is strong*.)

However, there is no harm to consider that "*misoshiru (miso soup)*, *koi-me (strong)*" is also correct. If we judge these cases as correct, the Proposed models achieve nearly 0.8 precision and 0.7 recall, and the baseline model also get 7 % improvement (precision 0.63 and recall 0.6). Based on this result, we consider that we achieved reasonable performance in intra-sentential aspect-evaluation relation extraction.

As Table 3 shows, inter-sentential relation extraction achieved very poorly. In the case of inter-sentential relations, our model tends to rely heavily on the statistical clues, because syntactic pattern features cannot be used. However, our current method for estimating co-occurrence distributions is not sophisticated as we discussed above. We need to seek for more effective use of large scale domain dependent data to obtain better statistics.

We also conducted a preliminary test of the opinion-hood determination model using the features used in aspect-evaluation relation extraction. As a result, we got 0.5 precision and 0.45 recall. Opinion-hood determination problem includes two decisions: whether the evaluation candidate is an opinion or not, and whether the opinion is related to the given domain if the evaluation candidate is an opinion. We plan to use various features known to be effective in the sentence subjectivity recognition task. This task involves challenging problems. For example, sentence (1) includes the writer's evaluation on *shrimps* served at a particular restaurant. In contrast, very similar sentence (2) does not express evaluation since it is a generic description of the writer's taste.

(1) *watashi-wa konomise-no ebi-ga suki-desu*
I the restaurant shrimp like
(*I like shrimps of the restaurant*.)

(2) *watashi-wa ebi-ga suki-desu*
I shrimps like
(*I like shrimps*.)

Thus we need to conduct further investigation in order to resolve this kind of problems.

4.4 Portability of intra-sentential model

We next evaluated effectiveness of the contextual clues learned in the domains to other domains by testing a model trained on the certain domains to other domain. We selected two new domains, cellular phone and automobile, and annotated 290 weblog posts in each domain. For the restaurant domain, we randomly selected 290 posts from the previously mentioned our annotated corpus. We then divide each data set to a training set and a test set so that we had the same amount of training data for each domain. Then we trained a model on the data for each domain, and applied it to each of the three set of data. Table 5 shows the results of the experiment. Compared with the model trained on the same domain, the models trained on different domains exhibited almost comparable performance. This indicates that the contextual clues learned in other domains are effective in another domain, showing the cross-domain portability of our intra-sentential model.

Table 5: Comparing intra-sentential models among three domains (upper: aspect-eval, lower: aspect-of)

test		restaurant	cellular phone	automobile
same	P	0.72 (502/694)	0.75 (522/693)	0.76 (562/738)
dom.	R	0.62 (502/809)	0.63 (522/833)	0.65 (562/870)
other	P	0.73 (468/638)	0.72 (517/710)	0.74 (565/768)
dom	R	0.58 (468/809)	0.62 (517/833)	0.65 (565/870)
same	P	0.43 (139/321)	0.62 (139/224)	0.66 (185/280)
dom.	R	0.59 (139/234)	0.60 (139/230)	0.66 (185/279)
other	P	0.42 (124/293)	0.53 (138/260)	0.59 (195/329)
dom	R	0.52 (124/234)	0.60 (138/230)	0.70 (195/279)

5 Conclusion

In this paper, we described our opinion extraction task, which extract opinion units consisting of four constituents. We showed the feasibility of the task definition based on our corpus study. We consider the task as two kinds of relation extraction tasks, aspect-evaluation relation extraction and aspect-of relation extraction, and proposed a machine learning-based method which combines contextual clues and statistical clues. Our experimental results show that the model using contextual clues improved the performance for both tasks. We also showed domain portability of the contextual clues.

References

R. Bunescu. 2003. Associative anaphora resolution: a web-based approach. In *Proceedings of the EACL Workshop on The Computational Treatment of Anaphora*, pages 47–52.

Y. Choi, E. Breck, and C. Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 431–439.

H. H. Clark. 1977. *Bridging. Thinking: readings in cognitive science*. Cambridge : Cambridge University Press.

T. Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval* (*SIGIR*), pages 50–57.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining* (*KDD*), pages 168–177.

R. Iida, K. Inui, H. Takamura, and Y. Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the EACL Workshop on The Computational Treatment of Anaphora*, pages 23–30.

S. Ikehara, M. Miyazaki, S. Shirai A. Yokoo, H. Nakaiwa, K. Ogura, Y. Ooyama, and Y. Hayashi. 1997. *Nihongo Goi Taikei (in Japanese)*. Iwanami Shoten.

H. Kanayama and T. Nasukawa. 2004. Deeper sentiment analysis using machine translation technology. In *Proc. of the 20th International Conference on Computational Linguistics*(*COLING*), pages 494–500.

S. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th International Conference on Computational Linguistics* (*COLING*), pages 1367–1373.

S. Kim and E. Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the ACL Workshop on Sentiment and Subjectivity in Text*.

N. Kobayashi, K. Inui, Y. Matsumoto, K. Tateishi, and T. Fukushima. 2004. Collecting evaluative expressions for opinion extraction. In *Proceedings of the 1st International Joint Conference on Natural Language Processing* (*IJCNLP*) , pages 584–589.

N. Kobayashi, R. Iida, K. Inui, and Y. Matsumoto. 2005. Opinion extraction using a learning-based anaphora resolution technique. In *The Second International Joint Conference on Natural Language Processing* (*IJCNLP*), *Companion Volume to the Proceeding of Conference including Posters/Demos and Tutorial Abstracts*, pages 175–180.

T. Kudo and Y. Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP*).

B. Liu, M. Hu, and J. Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International World Wide Web Conference* (*WWW*), pages 342–351.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 79–86.

M. Poesio, R. Mehta, A. Maroudas, and J. Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

A. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 339–346.

Y. Suzuki, H. Takamura, and M. Okumura. 2006. Application of semi-supervised learning to evaluative expression classification. In *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics* (*CICLing*).

H. Takamura, T. Inui, and M. Okumura. 2006. Latent variable models for semantic orientations of phrases. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics* (*EACL*) , pages 201–208.

K. Tateishi, T. Fukushima, N. Kobayashi, T. Takahashi, A. Fujita, K. Inui, and Y. Matsumoto. 2004. Web opinion extraction and summarization based on viewpoints of products. In *IPSJ SIGNL Note 163*, pages 1–8. (in Japanese).

P. D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 417–424.

J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39:165–210.

J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the third IEEE International Conference on Data Mining* (*ICDM*), pages 427–434.

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 129–136.

Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents

Nobuhiro Kaji and **Masaru Kitsuregawa**
Institute of Industrial Science, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505 Japan
{kaji,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract

Recognizing polarity requires a list of polar words and phrases. For the purpose of building such lexicon automatically, a lot of studies have investigated (semi-) unsupervised method of learning polarity of words and phrases. In this paper, we explore to use structural clues that can extract polar sentences from Japanese HTML documents, and build lexicon from the extracted polar sentences. The key idea is to develop the structural clues so that it achieves extremely high precision at the cost of recall. In order to compensate for the low recall, we used massive collection of HTML documents. Thus, we could prepare enough polar sentence corpus.

1 Introduction

Sentiment analysis is a recent attempt to deal with evaluative aspects of text. In sentiment analysis, one fundamental problem is to recognize whether given text expresses positive or negative evaluation. Such property of text is called polarity. Recognizing polarity requires a list of polar words and phrases such as ’good’, ’bad’ and ’high performance’ etc. For the purpose of building such lexicon automatically, a lot of studies have investigated (semi-) unsupervised approach.

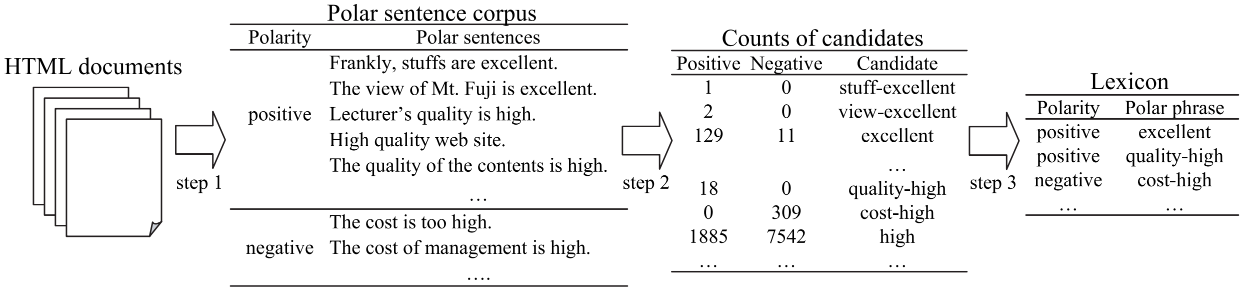
So far, two kinds of approaches have been proposed to this problem. One is based on a thesaurus. This method utilizes synonyms or glosses of a thesaurus in order to determine polarity of words (Kamps et al., 2004; Hu and Liu, 2004; Kim and Hovy, 2004; Esuli and Sebastiani, 2005). The second approach exploits raw corpus. Polarity is decided by using co-occurrence in a corpus. This is based on a hypothesis that polar phrases conveying the same polarity co-occur with each other. Typically, a small set of seed polar phrases are prepared, and new polar phrases are detected based on the strength of co-occurrence with the seeds (Hatzivassiloglous and McKeown, 1997; Turney, 2002; Kanayama and Nasukawa, 2006).

As for the second approach, it depends on the definition of co-occurrence whether the hypothesis is appropriate or not. In Turney’s work, the co-occurrence is considered as the appearance in the same window (Turney, 2002). Although this idea is simple and feasible, there is a room for improvement. According to Kanayama’s investigation, the hypothesis is appropriate in only 60% of cases if co-occurrence is defined as the appearance in the same window[1]. In Kanayama’s method, the co-occurrence is considered as the appearance in intra- or inter-sentential context (Kanayama and Nasukawa, 2006). They reported that the precision was boosted to 72.2%, but it is still not enough. Therefore, we think that the above hypothesis is often inappropriate in practice, and this fact is the biggest obstacle to learning lexicon from corpus.

In this paper, we explore to use structural clues that can extract polar sentences from Japanese HTML documents, and build lexicon from the ex-

[1] To be exact, the precision depends on window size and ranges from 59.7 to 64.1%. See Table 4 in (Kanayama and Nasukawa, 2006) for the detail.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1075–1083, Prague, June 2007. ©2007 Association for Computational Linguistics



Polar sentence corpus

Polarity	Polar sentences
positive	Frankly, stuffs are excellent.
	The view of Mt. Fuji is excellent.
	Lecturer's quality is high.
	High quality web site.
	The quality of the contents is high.
	…
negative	The cost is too high.
	The cost of management is high.
	….

Counts of candidates

Positive	Negative	Candidate
1	0	stuff-excellent
2	0	view-excellent
129	11	excellent
		…
18	0	quality-high
0	309	cost-high
1885	7542	high
…	…	…

Lexicon

Polarity	Polar phrase
positive	excellent
positive	quality-high
negative	cost-high
…	…

Figure 1: Overview of the proposed method.

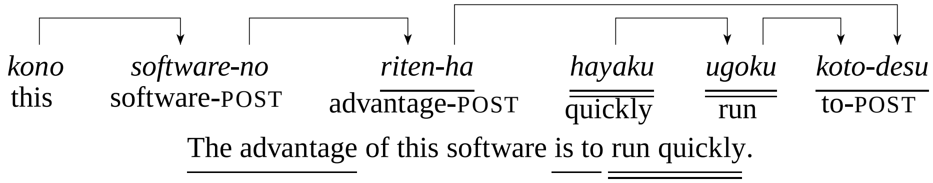


Figure 2: Language structure.

tracted polar sentences. An overview of the proposed method is represented in Figure 1. First, polar sentences are extracted from HTML documents by using structural clues (step 1). The set of polar sentences is called polar sentence corpus. Next, from the polar sentence corpus, candidates of polar phrases are extracted together with their counts in positive and negative sentences (step 2). Finally, polar phrases are selected from the candidates and added to our lexicon (step 3).

The key idea is to develop the structural clues so that it achieves extremely high precision at the cost of recall. As we will see in Section 2.3, the precision was extremely high. It was around 92% even if ambiguous cases were considered as incorrect. In order to compensate for the low recall, we used massive collection of HTML documents. Thus, we could build enough polar sentence corpus. To be specific, we extracted 500,000 polar sentences from one billion HTML documents.

The contribution of this paper is to empirically show the effectiveness of an approach that makes use of the strength of massive data. Nowadays, terabyte is not surprisingly large, and larger corpus would be obtained in the future. Therefore, we think this kind of research direction is important.

2 Extracting Polar Sentences

Our method begins by automatically constructing polar sentence corpus with structural clues (step 1).

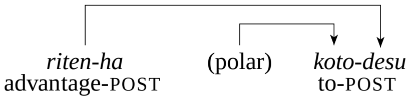
The basic idea is exploiting certain language and layout structures as clues to extract polar sentences. The clues were carefully chosen so that it achieves high precision. The original idea was represented in our previous paper (Kaji and Kitsuregawa, 2006).

2.1 Language structure

Some polar sentences are described by using characteristic language structures. Figure 2 illustrates such Japanese polar sentence attached with English translations. Japanese are written in italics and '-' denotes that the word is followed by postpositional particles. For example, '*software-no*' means that '*software*' is followed by postpositional particle '*no*'. The arrow represents dependency relationship. Translations are shown below the Japanese sentence. '-POST' means postpositional particle.

What characterizes this sentence is the singly underlined phrase. In this phrase, '*riten* (advantage)' is followed by postpositional particle '*-ha*', which is Japanese topic marker. And hence, we can recognize that something positive is the topic of the sentence. This kind of linguistic structure can be recognized by lexico-syntactic pattern. Hereafter, such words like '*riten* (advantage)' are called cue words.

In order to handle the language structures, we utilized lexico-syntactic patterns as illustrated below.



A sub-tree that matches (polar) is extracted as polar sentence. It is obvious whether the polar sentence is positive or negative one. In case of Figure 2, the doubly underlined part is extracted as polar sentence[2].

Besides '*riten* (advantage)', other cue words were also used. A list of cue words (and phrases) were manually created. For example, we used 'pros' or 'good point' for positive sentences, and 'cons', 'bad point' or 'disadvantage' for negative ones. This list is also used when dealing with layout structures.

2.2 Layout structure

Two kinds of layout structures are utilized as clues. The first clue is the itemization. In Figure 3, the itemizations have headers and they are cue words ('pros' and 'cons'). Note that we illustrated translations for the sake of readability. By using the cue words, we can recognize that polar sentences are described in these itemizations.

The other clue is table structure. In Figure 4, a car review is summarized in the table format. The left column acts as a header and there are cue words ('plus' and 'minus') in that column.

Pros:

- The sound is natural.
- Music is easy to find.
- Can enjoy creating my favorite play-lists.

Cons:

- The remote controller does not have an LCD display.
- The body gets scratched and fingerprinted easily.
- The battery drains quickly when using the backlight.

Figure 3: Itemization structure.

Mileage(urban)	7.0km/litter
Mileage(highway)	9.0km/litter
Plus	This is a four door car, but it's so cool.
Minus	The seat is ragged and the light is dark.

Figure 4: Table structure.

It is easy to extract polar sentences from the itemization. Such itemizations as illustrated in Figure 3 can be detected by using the list of cue words and HTML tags such as <h1> and <ul> etc. Three positive and negative sentences are extracted respectively from Figure 3.

As for table structures, two kinds of tables are considered (Figure 5). In the Figure, $+$ and $-$ represent positive and negative polar sentences, and C_+ and C_- represent cue words. Type A is a table in which the leftmost column acts as a header. Figure 4 is categorized into this type. Type B is a table in which the first row acts as a header.

Type A

C_+	$+$
C_-	$-$

Type B

	C_+	C_-	
	$+$	$-$	

Figure 5: Two types of table structures.

In order to extract polar sentences, first of all, it is necessary to determine the type of the table. The table is categorized into type A if there are cue words in the leftmost column. The table is categorized into type B if it is not type A and there are cue words in the first row. After the type of the table is decided, we can extract polar sentences from the cells that correspond to $+$ and $-$ in the Figure 5.

2.3 Result of corpus construction

The method was applied to one billion HTML documents. In order to get dependency tree, we used KNP[3]. As the result, 509,471 unique polar sentences were obtained. 220,716 are positive and the others are negative[4]. Table 1 illustrates some translations of the polar sentences.

[2] To be exact, the doubly underlined part is polar clause. However, it is called polar sentence because of the consistency with polar sentences extracted by using layout structures.

[3] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp.html

[4] The polar sentence corpus is available from http://www.tkl.iis.u-tokyo.ac.jp/~kaji/acp/.

Table 1: Examples of polar sentences.

Polarity	Polar sentence
positive	It becomes easy to compute cost. It's easy and can save time. The soup is rich and flavorful.
negative	Cannot use mails in HTML format. The lecture is really boring. There is no impressive music.

In order to investigate the quality of the corpus, two human judges (judge A/B) assessed 500 polar sentences in the corpus. According to the judge A, the precision was 91.4%. 459 out of 500 polar sentences were regarded as valid ones. According to the judge B, the precision was 92.0% (460/500). The agreement between the two judges was 93.5% (Kappa value was 0.90), and thus we can conclude that the polar sentence corpus has enough quality (Kaji and Kitsuregawa, 2006).

After error analysis, we found that most of the errors are caused by the lack of context. The following is a typical example.

> There is much information.

This sentence is categorized into positive one in the corpus, and it was regarded as invalid by both judges because the polarity of this sentence is ambiguous without context.

As we described in Section 1, the hypothesis of co-occurrence based method is often inappropriate. (Kanayama and Nasukawa, 2006) reported that it was appropriate in 72.2% of cases. On the other hand, by using extremely precise clues, we could build polar sentence corpus that have high precision (around 92%). Although the recall of structural clues is low, we could build large corpus by using massive collection of HTML documents. Of course, we cannot directly compare these two percentages. We think, however, the high precision of 92% implies the strength of our approach.

3 Acquisition of Polar Phrases

The next step is to acquire polar phrases from the polar sentence corpus (step 2 and 3 in Figure 1).

3.1 Counting candidates

From the corpus, candidates of polar phrases are extracted together with their counts (step 2).

As is often pointed out, adjectives are often used to express evaluative content. Considering that polarity of isolate adjective is sometimes ambiguous (e.g. high), not only adjectives but also adjective phrases (noun + postpositional particle + adjective) are treated as candidates. Adjective phrases are extracted by the dependency parser. To handle negation, an adjective with negation words such as 'not' is annotated by <NEGATION> tag. For the sake of readability, we simply represent adjective phrases in the form of 'noun-adjective' by omiting postpositional particle, as in the Figure 1.

For each candidate, we count the frequency in positive and negative sentences separately. Intuitively, we can expect that positive phrases often appear in positive sentences, and vice versa. However, there are exceptional cases as follows.

> Although the price is high, its shape is beautiful.

Although this sentence as a whole expresses positive evaluation and it is positive sentence, negative phrase 'price is high' appears in it. To handle this, we hypothesized that positive/negative phrases tend to appear in main clause of positive/negative sentences, and we exploited only main clauses to count the frequency.

3.2 Selecting polar phrases

For each candidate, we determine numerical value indicating the strength of polarity, which is referred as polarity value. On the basis of this value, we select polar phrases from the candidates and add them to our lexicon (step 3).

For each candidate c, we can create a contingency table as follows.

Table 2: Contingency table

	pos	neg
c	$f(c, pos)$	$f(c, neg)$
$\neg c$	$f(\neg c, pos)$	$f(\neg c, neg)$

$f(c, pos)$ is the frequency of c in positive sentences. $f(\neg c, pos)$ is that of all candidates but c. $f(c, neg)$ and $f(\neg c, neg)$ are similarly decided.

From this contingency table, c's polarity value is determined. Two ideas are examined for comparison. One is based on chi-square value and the other is based on Pointwise Mutual Information (PMI).

Chi-square based polarity value The chi-square value is a statistical measure used to test the null hypothesis that, in our case, the probability of a candidate in positive sentences is equal to the probability in negative sentences. Given Table 2, the chi-square value is calculated as follows.

$$\chi^2(c) = \sum_{x \in (c, \neg c)} \sum_{y \in (pos, neg)} \frac{\{f(x,y) - \hat{f}(x,y)\}^2}{\hat{f}(x,y)}$$

Here, $\hat{f}(x,y)$ is the expected value of $f(x,y)$ under the null hypothesis.

Although $\chi^2(c)(\geq 0)$ indicates the strength of bias toward positive or negative sentences, its direction is not clear. We determined polarity value so that it is greater than zero if c appears in positive sentences more frequently than in negative sentences and otherwise it is less than zero.

$$PV_{\chi^2}(c) = \begin{cases} \chi^2(c) & \text{if} P(c|neg) < P(c|pos) \\ -\chi^2(c) & \text{otherwise} \end{cases}$$

$P(c|pos)$ is c's probability in positive sentences, and $P(c|neg)$ is that in negative sentences. They are estimated by using Table 2.

$$\begin{aligned} P(c|pos) &= \frac{f(c,pos)}{f(c,pos) + f(\neg c, pos)} \\ P(c|neg) &= \frac{f(c,neg)}{f(c,neg) + f(\neg c, neg)} \end{aligned}$$

PMI based polarity value Using PMI, the strength of association between c and positive sentences (and negative sentences) is defined as follows (Church and Hanks, 1989).

$$\begin{aligned} PMI(c,pos) &= \log_2 \frac{P(c,pos)}{P(c)P(pos)} \\ PMI(c,neg) &= \log_2 \frac{P(c,neg)}{P(c)P(neg)} \end{aligned}$$

PMI based polarity value is defined as their difference. This idea is the same as (Turney, 2002).

$$\begin{aligned} PV_{PMI}(c) &= PMI(c,pos) - PMI(c,neg) \\ &= \log_2 \frac{P(c,pos)/P(pos)}{P(c,neg)/P(neg)} \\ &= \log_2 \frac{P(c|pos)}{P(c|neg)} \end{aligned}$$

$P(c|pos)$ and $P(c|neg)$ are estimated in the same way as shown above. $PV_{PMI}(c)$ is (log of) the ratio of c's probability in positive sentences to that in negative sentences. This formalization follows our intuition. Similar to $PV_{\chi^2}(c)$, $PV_{PMI}(c)$ is greater than zero if $P(c|neg) < P(c|pos)$, otherwise it is less than zero.

Selecting polar phrases By using polarity value and threshold $\theta(> 0)$, it is decided whether a candidate c is polar phrase or not. If $\theta < PV(c)$, the candidate is regarded as positive phrase. Similarly, if $PV(c) < -\theta$, it is regarded as negative phrase. Otherwise, it is regarded as neutral. Only positive and negative phrases are added to our lexicon. By changing θ, the trade-off between precision and recall can be adjusted. In order to avoid data sparseness problem, if both $f(c,pos)$ and $f(c,neg)$ are less than three, such candidates were ignored.

4 Related Work

As described in Section 1, there have been two approaches to (semi-) unsupervised learning of polarity. This Section introduces the two approaches and other related work.

4.1 Thesaurus based approach

Kamps et al. built lexical network by linking synonyms provided by a thesaurus, and polarity was defined by the distance from seed words ('good' and 'bad') in the network (Kamps et al., 2004). This method relies on a hypothesis that synonyms have the same polarity. Hu and Liu used similar lexical network, but they considered not only synonyms but antonyms (Hu and Liu, 2004). Kim and Hovy proposed two probabilistic models to estimate the strength of polarity (Kim and Hovy, 2004). In their models, synonyms are used as features. Esuli et al. utilized glosses of words to determine polarity (Esuli and Sebastiani, 2005; Esuli and Sebastiani, 2006).

Compared with our approach, the drawback of using thesaurus is the lack of scalability. It is difficult to handle such words that are not contained in a thesaurus (e.g. newly-coined words or colloquial words). In addition, phrases cannot be handled because the entry of usual thesaurus is not phrase but word.

4.2 Corpus based approach

Another approach is based on an idea that polar phrases conveying the same polarity co-occur with each other in corpus.

(Turney, 2002) is one of the most famous work that discussed learning polarity from corpus. Turney determined polarity value[5] based on co-occurrence with seed words ('excellent' and 'poor'). The co-occurrence is measured by the number of hits returned by a search engine. The polarity value proposed by (Turney, 2002) is as follows.

$$\log_2 \frac{\mathrm{hits}(c \,\mathbf{NEAR}\, \mathrm{excellent})\mathrm{hits}(\mathrm{poor})}{\mathrm{hits}(c \,\mathbf{NEAR}\, \mathrm{poor})\mathrm{hits}(\mathrm{excellent})}$$

$\mathrm{hits}(q)$ means the number of hits returned by a search engine when query q is issued. $\mathbf{NEAR}$ means NEAR operator, which enables to retrieve only such documents that contain two queries within ten words.

Hatzivassiloglou and McKeown constructed lexical network and determine polarity of adjectives (Hatzivassiloglous and McKeown, 1997). Although this is similar to thesaurus based approach, they built the network from intra-sentential co-occurrence. Takamura et al. built lexical network from not only such co-occurrence but other resources including thesaurus (Takamura et al., 2005). They used spin model to predict polarity of words.

Popescu and Etzioni applied relaxation labeling to polarity identification (Popescu and Etzioni, 2005). This method iteratively assigns polarity to words by using various features including intra-sentential co-occurrence and synonyms of a thesaurus.

Kanayama and Nasukawa used both intra- and inter-sentential co-occurrence to learn polarity of words and phrases (Kanayama and Nasukawa, 2006). Their method covers wider range of co-occurrence than other work such as (Hatzivassiloglous and McKeown, 1997). An interesting point of this work is that they discussed building domain oriented lexicon. This is contrastive to other work including ours that addresses to build domain independent lexicon.

In summary, the strength of our approach is to exploit extremely precise structural clues, and to use massive collection of HTML documents to compensate for the low recall. Although Turney's method also uses massive collection of HTML documents, his method does not make much of precision compared with our method. As we will see in Section 5, our experimental result revealed that our method overwhelms Turney's method.

4.3 Other related work

In some review sites, pros and cons are stated using such layout that we introduced in Section 2. Some work examined the importance of such layout (Liu et al., 2005; Kim and Hovy, 2006). However, they regarded layout structures as clues specific to a certain review site. They did not propose to use layout structure to extract polar sentences from arbitrary HTML documents.

Some studies addressed supervised approach to learning polarity of phrases (Wilson et al., 2005; Takamura et al., 2006). These are different from ours in a sense that they require manually tagged data.

Kobayashi et al. proposed a framework to reduce the cost of manually building lexicon (Kobayashi et al., 2004). In the experiment, they compared the framework with fully manual method and investigated the effectiveness.

5 Experiment

A test set consisting of 405 adjective phrases were created. From the test set, we extract polar phrases by looking up our lexicon. The result was evaluated through precision and recall[6].

5.1 Setting

The test set was created in the following manner. 500 adjective phrases were randomly extracted from the Web text. Note that there is no overlap between our polar sentence corpus and this text. After removing parsing error and duplicates, 405 unique adjective phrases were obtained. Each phase was manually annotated with polarity tag (positive, negative and neutral), and we obtained 158 positive phrases, 150 negative phrases and 97 neutral phrases. In order to check the reliability of annotation, another

[5] Semantic Orientation in (Turney, 2002).

[6] The lexicon is available from http://www.tkl.iis.u-tokyo.ac.jp/~kaji/polardic/.

Table 3: The experimental result (chi-square).

θ		0	10	20	30	40	50	60
Precision/Recall	Positive	76.4/92.4	84.0/86.7	84.1/83.5	86.2/79.1	88.7/74.7	86.7/65.8	86.7/65.8
	Negative	68.5/84.0	65.5/63.3	64.3/60.0	62.7/57.3	81.1/51.3	80.0/48.0	80.0/48.0
# of polar words and phrases		9,670	2,056	1,047	698	533	423	335

Table 4: The experimental result (PMI).

θ		0	0.5	1.0	1.5	2.0	2.5	3.0
Precision/Recall	Positive	76.4/92.4	79.6/91.1	86.1/89.9	87.2/86.1	90.9/82.3	92.4/76.6	92.9/65.8
	Negative	68.5/84.0	75.8/81.3	82.3/77.3	84.8/74.7	85.8/72.7	86.8/70.0	87.9/62.7
# of polar words and phrases		9,670	9,320	9,039	8,804	8,570	8,398	8,166

Table 5: The effect of data size (PMI, θ=1.0).

size		1/20	1/15	1/10	1/5	1
Precision/Recall	Positive	87.0/63.9	84.6/65.8	85.1/75.9	85.4/84.8	86.1/89.9
	Negative	76.9/55.8	86.2/50.0	82.1/58.0	80.3/62.7	82.3/77.3

human judge annotated the same data. The Kappa value between the two judges was 0.73, and we think the annotation is reliable.

From the test set, we extracted polar phrases by looking up our lexicon. As for adjectives in the lexicon, partial match is allowed. For example, if the lexicon contains an adjective ’excellent’, it matches every adjective phrase that includes ’excellent’ such as ’view-excellent’ etc.

As a baseline, we built lexicon similarly by using polarity value of (Turney, 2002). As seed words, we used ’*saikou* (best)’ and ’*saitei* (worst)’. Some seeds were tested and these words achieved the best result. As a search engine, we tested Google and our local engine, which indexes 150 millions Japanese documents. Its size is compatible to (Turney and Littman, 2002). Since Google does not support NEAR, we used AND. Our local engine supports NEAR.

5.2 Results and discussion

We evaluated the result of polar phrase extraction. By changing the threshold θ, we investigated recall-precision curve (Figure 6 and 7). The detail is represented in Table 3 and 4. The second/third row represents precision and recall of positive/negative phrases. The fourth row is the size of the lexicon.

The Figures show that both of the proposed methods outperform the baselines. The best F-measure was achieved by PMI (θ=1.0). Although Turney’s method may be improved with minor configurations (e.g. using other seeds etc.), we think this results indicate the feasibility of the proposed method. Although the size of lexicon is not surprisingly large, it would be possible to make the lexicon larger by using more HTML documents. In addition, notice that we focus on only adjectives and adjective phrases.

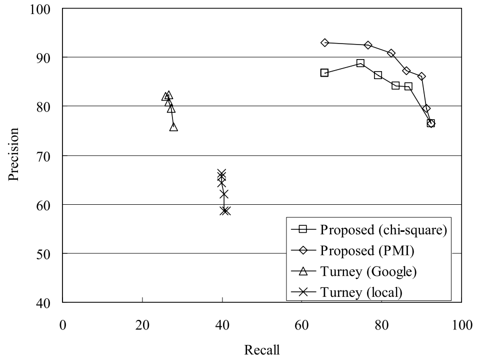


Figure 6: Recall-precision curve (positive phrases)

Comparing the two proposed methods, PMI is always better than chi-square. Especially, chi-square suffers from low recall, because the size of lexicon is extremely small. For example, when the threshold is 60, the precision is 80% and the recall is 48% for negative phrases. On the other hand, PMI would achieve the same precision when recall is around 80% (θ is between 0.5 and 1.0).

Turney’s method did not work well although they reported 80% accuracy in (Turney and Littman, 2002). This is probably because our experimental setting is different. Turney examined binary classification of positive and negative words, and we discussed extracting positive and negative phrases from the set of positive, negative and neutral phrases.

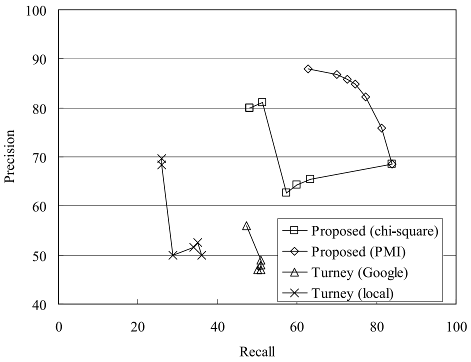


Figure 7: Recall-precision curve (negative phrases)

Error analysis revealed that most of the errors are related to neutral phrases. For example, PMI (θ=1.0) extracted 48 incorrect polar phrases, and 37 of them were neutral phrases. We think one reason is that we did not use neutral corpus. It is one future work to exploit neutral corpus. The importance of neutral category is also discussed in other literatures (Esuli and Sebastiani, 2006).

To further assess our method, we did two additional experiments. In the first experiment, to investigate the effect of data size, the same experiment was conducted using 1/n (n=1,5,10,15,20) of the entire polar sentence corpus (Table 5). PMI (θ=1.0) was also used. As the size of corpus increases, the performance becomes higher. Especially, the recall is improved dramatically. Therefore, the recall would be further improved using more corpus.

In the other experiment, the lexicon was evaluated directly so that we can examine polar words and phrases that are not in the test set. We think it is difficult to fully assess low frequency words in the previous setting. Two human judges assessed 200 unique polar words and phrases in the lexicon (PMI, θ=1.0). The average precision was 71.3% (Kappa value was 0.66). The precision is lower than the result in Table 4. This result indicates that it is difficult to handle low frequency words.

The Table 6 illustrates examples of polar phrases and their polarity values. We can see that both phrases and colloquial words such as 'uncool' are appropriately learned. They are difficult to handle for thesaurus based approach, because such words are not usually in thesaurus.

Table 6: Examples

polar phrase	$PV_{\chi^2}(c)$	$PV_{PMI}(c)$
kenkyoda (modest)	38.3	12.1
exiting (exiting)	13.5	10.4
more-sukunai (leak-small)	9.2	9.8
dasai (uncool)	-2.9	-3.3
yakkaida (annoying)	-11.9	-3.9
shomo-hayai (consumption-quick)	-17.7	-4.4

It is important to discuss how general our framework is. Although the lexico-syntactic patterns shown in Section 2 are specific to Japanese, we think that the idea of exploiting language structure is applicable to other languages including English. Roughly speaking, the pattern we exploited can be translated into 'the advantage/weakness of something is to ...' in English. It is worth pointing out that lexico-syntactic patterns have been widely used in English lexical acquisition (Hearst, 1992). Obviously, other parts of the proposed method does not depend on Japanese.

6 Conclusion

In this paper, we explore to use structural clues that can extract polar sentences from Japanese HTML documents, and build lexicon from the extracted polar sentences. The key idea is to develop the structural clues so that it achieves extremely high precision at the cost of recall. In order to compensate for the low recall, we used massive collection of HTML documents. Thus, we could prepare enough polar sentence corpus. Experimental result demonstrated the feasibility of our approach.

Acknowledgement This work was supported by the *Comprehensive Development of e-Society Foundation Software* program of the Ministry of Education, Culture, Sports, Science and Technology, Japan. We would like to thank Assistant Researcher Takayuki Tamura for his development of the Web crawler.

References

Kenneth Ward Church and Patric Hanks. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of ACL*, pages 76–83.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms throush gloss classification. In *Proceedings of CIKM*, pages 617–624.

Andrea Esuli and Fabrizio Sebastiani. 2006. Determining term subjectivity and term orientation for opinion mining. In *Proceedings of EACL*, pages 193–200.

Vasileios Hatzivassiloglous and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of ACL*, pages 174–181.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*, pages 539–545.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*, pages 168–177.

Nobuhiro Kaji and Masaru Kitsuregawa. 2006. Automatic construction of polarity-tagged corpus from html documents. In *Proceedings of COLING/ACL, poster sessions*, pages 452–459.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten de Rijke. 2004. Using wordnet to measure semantic orientations of adjectives. In *Proceedings of LREC*.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of ENMLP*, pages 355–363.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING*, pages 1367–1373.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of COLING/ACL Poster Sessions*, pages 483–490.

Nozomi Kobayashi, Kentaro Inui, Yuji Matsumoto, Kenji Tateishi, and Toshikazu Fukushima. 2004. Collecting evaluative expressions for opinion extraction. In *Proceedings of IJCNLP*, pages 584–589.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW*.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientation of words using spin model. In *Proceedings of ACL*, pages 133–140.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2006. Latent variable mdels for semantic orientations of phrases. In *Proceedings of EACL*, pages 201–208.

Peter D. Turney and Michael L. Littman. 2002. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. Technical report, National Research Council Canada.

Peter D. Turney. 2002. Thumbs up or thumbs down ? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL*, pages 417–424.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP*.

Determining Case in Arabic: Learning Complex Linguistic Behavior Requires Complex Linguistic Features

Nizar Habash†, Ryan Gabbard‡, Owen Rambow†, Seth Kulick‡ and Mitch Marcus‡

†Center for Computational Learning Systems, Columbia University
New York, NY, USA
{habash,rambow}@cs.columbia.edu
‡Department of Computer and Information Science,University of Pennsylvania
Philadelphia, PA, USA
{gabbard,skulick,mitch}@cis.upenn.edu

Abstract

This paper discusses automatic determination of case in Arabic. This task is a major source of errors in full diacritization of Arabic. We use a gold-standard syntactic tree, and obtain an error rate of about 4.2%, with a machine learning based system outperforming a system using hand-written rules. A careful error analysis suggests that when we account for annotation errors in the gold standard, the error rate drops to 0.8%, with the hand-written rules outperforming the machine learning-based system.

1 Introduction

In Modern Standard Arabic (MSA), all nouns and adjectives have one of three cases: nominative (NOM), accusative (ACC), or genitive (GEN). What sets case in MSA apart from case in other languages is most saliently the fact that it is usually not marked in the orthography, as it is written using diacritics which are normally omitted. In fact, in a recent paper on diacritization, Habash and Rambow (2007) report that word error rate drops 9.4% absolute (to 5.5%) if the word-final diacritics (which include case) need not be predicted. Similar drops have been observed by other researchers (Nelken and Shieber, 2005; Zitouni et al., 2006). Thus, we can deduce that tagging-based approaches to case identification are limited in their usefulness, and if we need full diacritization for subsequent processing in a natural language processing (NLP) application (say, language modeling for automatic speech recognition (Vergyri and Kirchhoff, 2004)), we need to perform more complex syntactic processing to restore case diacritics. Options include using the output of a parser in determining case.

An additional motivation for investigating case in Arabic comes from treebanking. Native speakers of Arabic in fact are native speakers of one of the Arabic dialects, all of which have lost case (Holes, 2004). They learn MSA in school, and have no native-speaker intuition about case. Thus, determining case in MSA is a hard problem for everyone, including treebank annotators. A tool to catch case-related errors in treebanking would be useful.

In this paper, we investigate the problem of determining case of nouns and adjectives in syntactic trees. We use gold standard trees from the Arabic Treebank (ATB). We see our work using gold standard trees as a first step towards developing a system for restoring case to the output of a parser. The complexity of the task justifies an initial investigation based on gold standard trees. And of course, the use of gold standard trees is justified for our other objective, helping quality control for treebanking.

The study presented in this paper shows the importance of what has been called “feature engineering” and the issue of representation for machine learning. Our initial machine learning experiments use features that can be read off the ATB phrase structure trees in a straightforward manner. The literature on case in MSA (prescriptive and descriptive sources) reveals that case assignment in Arabic does not always follow standard assumptions about predicate-argument structure, which is what

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1084–1092, Prague, June 2007. ©2007 Association for Computational Linguistics

the ATB annotation is based on. Therefore, we transform the ATB so that the new representation is based entirely on case assignment, not predicate-argument structure. The features for machine learning that can now be read off from the new representation yield much better results. Our results show that we can determine case with an error rate of 4.2%. However, our results would have been impossible without a deeper understanding of the linguistic phenomenon of case and a transformation of the representation oriented towards this phenomenon.

Using either underlying representation, machine learning performs better than hand-written rules. However, a closer look at the errors made by the machine learning-derived classifier and the hand-written rules reveals that most errors are in fact treebank errors (between 69% and 86% of all errors for the machine learning-derived classifier and the hand-written rules, respectively). Furthermore, the machine learning classifier agrees more often with treebank errors than the hand-written rules do. This fact highlights the problem of machine learning (garbage in, garbage out), but holds out the prospect for improvement in the machine learning based classifier as the treebank is checked for errors and re-released.

In the next section, we describe all relevant linguistic facts of case in Arabic. Section 3 details the resources used in this research. Section 4 describes the preprocessing done to extract the relevant linguistic features from the ATB. Sections 5 and 6 detail the two systems we compare. Sections 7 and 8 present results and an error analysis of the two systems. And we conclude with a discussion of our findings in Section 9.

2 Linguistic Facts

All Arabic nominals (common nouns, proper nouns, adjectives and adverbs) are inflected for case, which has three values in Arabic: nominative (NOM), accusative (ACC) or genitive (GEN). We know this from case agreement facts, even though the morphology and/or orthography do not necessarily always make the case realization overt. We discuss morphological and syntactic aspects of case in MSA in turn.

2.1 Morphological Realization of Case

The realization of nominal case in Arabic is complicated by its orthography, which uses optional diacritics to indicate short vowel case morphemes, and by its morphology, which does not always distinguish between all cases. Additionally, case realization in Arabic interacts heavily with the realization of definiteness, leading to different realizations depending on whether the nominal is indefinite, i.e., receiving *nunation* (تنوين), definite through the determiner *Al+* (+ال) or definite through being the governor of an idafa possessive construction (إضافة). Most details of this interaction are outside the scope of this paper, but we discuss it as much as it helps clarify issues of case.

Buckley (2004) describes eight different classes of nominal case expression, which we briefly review. We first discuss the realization of case in morphologically singular nouns (including broken, i.e., irregular, plurals). *Triptotes* are the basic class which expresses the three cases in the singular using the three short vowels of Arabic: NOM is ـُ +*u*,[1] ACC is ـَ +*a*, and GEN is ـِ +*i*. The corresponding nunated forms for these three diacritics are: ـٌ +*ũ* for NOM, ـً +*ã* for ACC, and ـٍ +*ĩ* for GEN. Nominals not ending with Ta Marbuta (ة *ħ*) or Alif Hamza (اء *A'*) receive an extra Alif in the accusative indefinite case (e.g, كِتابًا *kitAbAã* 'book' versus كِتابَةً *kitAbaħã* 'writing').

Diptotes are like triptotes except that when they are indefinite, they do not express nunation and they use the ـَ +*a* suffix for both ACC and GEN. The class of diptotes is lexically specific. It includes nominals with specific meanings or morphological patterns (colors, elatives, specific broken plurals, some proper names with Ta Marbuta ending or location names devoid of the definite article). Examples include بيروت *bayruwt* 'Beirut' and أزرق *Âazraq*

[1] All Arabic transliterations are provided in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007). This scheme extends Buckwalter's transliteration scheme (Buckwalter, 2002) to increase its readability while maintaining the 1-to-1 correspondence with Arabic orthography as represented in standard encodings of Arabic, i.e., Unicode, CP-1256, etc. The following are the only differences from Buckwalter's scheme (which is indicated in parentheses): Ā آ (|), Â أ (>), ŵ ؤ (&), Ǎ إ (<), ŷ ئ ({), *ħ* ة (p), *θ* ث (v), ð ذ (*), š ش ($), Ď ظ (Z), ς ع (E), *γ* غ (g), ý ى (Y), ã ـً (F), ũ ـٌ (N), ĩ ـٍ (K).

'blue'.

The next three classes are less common. The *invariables* show no case in the singular (e.g. nominals ending in long vowels: سوريا *suwryA* 'Syria' or ذكرى *ðikraý* 'memoir'). The *indeclinables* always use the ـَ +*a* suffix to express case in the singular and allow for nunation (معنًى *maςnaýã* 'meaning'). The *defective* nominals, which are derived from roots with a final radical glide (*y* or *w*), look like triptotes except that they collapse Nom and Gen into the Gen form, which also includes loosing their final glide: قاضٍ *qADĩ* (Nom,Gen) versus قاضيًا *qADiyAã* (Acc) 'a judge'.

For the dual and sound plural, the situation is simpler, as there are no lexical exceptions. The duals and masculine sound plurals express number, case and gender jointly in single morphemes that are identifiable even if undiacritized: كاتِبُونَ *kAtib+uwna* 'writers$_{masc,pl}$' (Nom), كاتِبَانِ *kAtib+Ani* 'writers$_{masc,du}$' (Nom), كاتِبَتَانِ *kAtib+atAni* 'writers$_{fem,du}$' (Nom). The Acc and Gen forms are identical, e.g., كاتِبِينَ *kAtib+iyna* 'writers$_{masc,pl}$' (Acc,Gen). Finally, the dual and masculine sound plural do not express nunation. On the other hand, the feminine sound plural marks nunation explicitly, and all of its case morphemes are written only as diacritics, e.g., كاتِباتُ *kAtib+At+u* 'writers$_{fem,pl}$' (Nom).

2.2 Syntax of Case

Traditional Arabic grammar makes a distinction between *verbal clauses* (جمل فعلية) and *nominal clauses* (جمل آسمية). Verbal clauses are verb-initial sentences, and we (counter to the Arabic grammatical tradition) include copula-initial clauses in this group. The copula is كان *kAn* 'to be' or one of her sisters. Nominal clauses begin with a topic (which is always a nominal), and continue with a complement which is either a verbal clause, a nominal predicate, or a prepositional predicate. If the complement of a topic is a verbal clause, an inflectional subject morpheme or a resumptive object clitic pronoun replace the argument which has become the topic.

Arabic case system falls within the class of nominative-accusative languages (as opposed to ergative-absolutive languages). Some of the common behavior of case in Arabic with other languages includes:[2]

- Nom is assigned to subjects of verbal clauses, as well as other nominals in headings, titles and quotes.
- Acc is assigned to (direct and indirect) objects of verbal clauses, verbal nouns, or active participles; to subjects of small clauses governed by other verbs (i.e., "exceptional case marking" or "raising to object" contexts; we remain agnostic on the proper analysis); adverbs; and certain interjections, such as شكرًا *šukrAã* 'Thank you'.
- Gen is assigned to objects of prepositions and to possessors in idafa (possessive) construction.
- There is a distinction between case-by-assignment and case-by-agreement. In case-by-assignment, a specific case is assigned to a nominal by its case assigner; whereas in case-by-agreement, the modifying or conjoined nominal copies the case of its governor.

Arabic case differs from case in other languages in the following conditions, which relate to nominal clauses and numbers.

- The topic (independently of its grammatical function) is Acc if it follows the subordinating conjunction إِنّ *Ǎin∼a* (or any of her "sisters": لأَنّ *liÂan∼a*, كأَنّ *kaÂan∼a*, لكنّ *lakin∼a*, etc.). Otherwise, the topic is Nom.
- Nominal predicates are Acc if they are governed by the overt copula. They are also Acc if they are objects of verbs that take small clause complements (such as 'to consider'), unless the predicate is introduced by a subordinating conjunction. In all other cases, they are Nom.
- In constructions involving a nominal and a number (عِشرُونَ كاتِبًا *Eišruwna kAtibAã* 'twenty writers'), the head of the phrase for case assignment is the number, which receives whichever case the context assigns. The case of the nominal depends on the number. If the number is between 11 and 99, the nominal is

[2] Buckley (2004) describes in detail the conditions for each of the three cases in Arabic. He considers Nom to be the default case. He specifies seven conditions for Nom, 25 for Acc and two for Gen. Our summary covers the same ground as his description except that we omit the vocative use of nominals.

ACC by *tamiyz* (تمييز – lit. “specification”). Otherwise, the nominal is GEN by idafa.

3 The Data

We use the third section of the current version of the Arabic Treebank released by the Linguistic Data Consortium (LDC) (Maamouri et al., 2004). We use the division into training and devtest corpora proposed by Zitouni et al. (2006), further dividing their devtest set into two equal parts to give us a development and a test set. The training set has approximately 367,000 words, and the development and test sets each have about 33,000 words. In our training data, of 133,250 case-marked nominals, 66.4% are GEN, 18.5% ACC, and 15.1% NOM.

The ATB annotation in principle indicates for each nominal its case and the corresponding realization (including diacritics). The only systematic exception is that invariables are not marked at all with their unrealized case, and are marked as having NOCASE. We exclude all nominals marked NOCASE from our evaluations, as we believe that these nominals actually do have case, it is just not marked in the treebank, and we do not wish to predict the morphological realization, only the underlying case. In reporting results, we use accuracy on the number of nominals whose case is given in the treebank.

While the ATB does not contain explicit information about headedness in its phrase structure, we can say that the syntactic annotations in the ATB are roughly based on predicate-argument structure. For example, for the structure shown in Figure 1, the “natural” interpretation is that the head is احتراقُ *AHtrAqu* ‘burning’, with a modifier منزلاً *mnzlAã* ‘house’, which in turn is modified by a QP whose head is (presumably) the number 20, which is modified by اكثرِ *Akθri* ‘more’ and من *mn* ‘than’. This dependency structure is shown on the left in Figure 2. Another annotation detail relevant to this paper is that the ATB marks the topic of a nominal clause as “SBJ” (i.e., as a subject) except when the predicate is a verbal clause; then it is marked as TPC. We consider these two cases to be the same case and relabel all such cases as TPC.

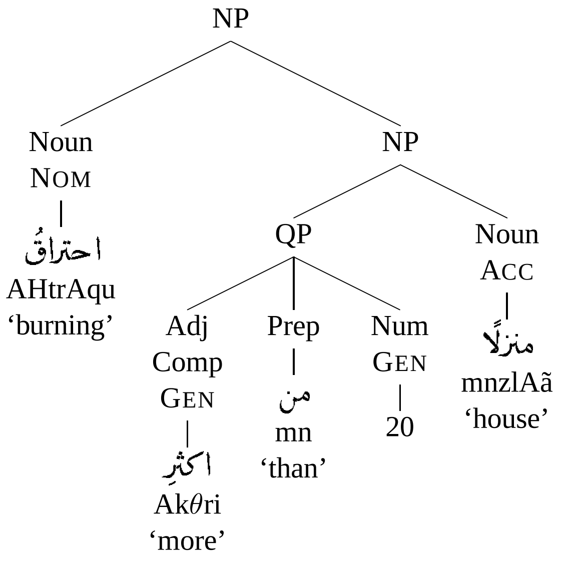


Figure 1: The representation of numbers in the Arabic Treebank, for a subject NP meaning ‘the burning of more than 20 houses’

4 Determining the Case Assigner

Case assignment is a relationship between two words: one word (the case governor or assigner) assigns case to the other word (the case assignee). Because case assignment is a relationship between words, we switch to a dependency-based version of the treebank. There are many possible ways to transform a phrase structure representation into a dependency representation; we explore two such conversions in the context of this paper. Note that if we had used the Prague Arabic Dependency Treebank (Smrž and Hajič, 2006) instead of the ATB, we would not have had to convert to dependency, but we still would have had to analyze whether the dependencies are the ones we need for modeling case assignment, possibly having to restructure the dependencies.

For determining the dependency relations that determine case assignment, we start out by using a standard head percolation algorithm with the following parameters: Verbs head all the arguments in VPs; prepositions head the PP arguments; and the first nominal in an NP or ADJP heads those structures. Non-verbal predicates (NPs, ADJPs or PPs) head their subjects (topics). The subordinating conjunction إنّ *Ǎin∼a* is governed by what follows it. The overt copula كان *kAn* governs both topic and predicate. Conjunctions are headed by what they follow and head what they precede (with the exception of the common sentence initial conjunction +و *w+* ‘and’, which is headed by the sentence it introduces). We will call the result of this algorithm the **Basic Case Assigner Identification Algorithm**, or **Basic Representation** for short.

After initial experiments with both hand-written rules and machine learning, we extend the Basic Representation in order to account for the special case assigning properties of numbers in Arabic by adding additional head percolation parameters and restructuring rules to handle the structure of NPs in the ATB. This is because the current ATB representation is not useful in some cases for representing case assignment. Consider the structure in Figure 1. Here, the head of the NP is the noun احتراقُ *AHtrAqu* ‘burning’, which has NOM because the NP is a subject (the verb is not shown). The QP’s first member, اكثرِ *Akθri* ‘more’ is GEN because it is in an idafa construction with the noun احتراقُ *AHtrAqu*. اكثرِ *Akθri* is modified by the preposition من *mn* ‘than’ which assigns GEN to the number 20 (which is written in Arabic numerals and thus does not show any case at all). The noun منزلاً *mnzlAã* ‘house’ is in a tamyiz relation with the number 20 which governs it, and thus it is ACC. It is clear that the phrase structure chosen for the ATB does not represent these case-assignment relations in a direct manner.

To create the appropriate head relations for case determination, we flatten all QPs and use a set of simple deterministic rules to create the more appropriate structure which expresses the chain of case assignments. In our development set, 5.8% of words get a new head using this new head assignment. We call this new representation the **Revised Representation**. Figure 2 shows the dependency representation corresponding to the phrase structure in Figure 1.

We make use of all dash-tags provided by the ATB as arc labels and we extend the label set to explicitly mark objects of prepositions (POBJ), possessors in idafa construction (IDAFA), conjuncts (CONJ) and conjunctions (CC), and the accusative specifier, tamyiz (TMZ). All other modifications receive the label (MOD).

5 Hand Written Rules

Our first system is based on hand-written rules (henceforth, we refer to this system as the rule-based system). We add two features to nominals in the tree: (1) we identify if a word governs a subordinating conjunction إِنّ *Ǎin∼a* or any of its sisters; and (2) we also identify if a topic of a nominal sentence has an *Ǎin∼a* sibling.

The following are the simple hand written rules we use:

- RULE 1: The default case assigned is ACC for all words.
- RULE 2: Assign NOM to nominals heading the tree and those labeled HLN (headline) or TTL (title).
- RULE 3: Assign GEN to nominals with the labels POBJ or IDAFA.
- RULE 4: Assign NOM to nominals with the label PRD if NOT headed by a verbal (verb or deverbal noun) or if it has an *Ǎin∼a* child.
- RULE 5: Assign NOM to nominal topics that do not have an *Ǎin∼a* sibling.
- RULE 6: All case-unassigned children of nominal parents (and conjunctions), whose label is MOD, CONJ or CC, copy the case of their parent. Conjunctions carry the case temporarily to pass on agreement. Verbs do not pass on agreement.

The first rule is applied to all nodes. The second to fifth rules are case-by-assignment rules applied in an if-else fashion (no overwriting is done). The last rule is a case-by-agreement rule. All non-nominals receive the case NA.

6 Machine Learning Experiments: The Statistical System

Our second system uses statistical machine learning. This system consists of a core model and an agreement model, both of which are linear classifiers trained using the maximum entropy technique. We implement this system using the MALLET toolbox (McCallum, 2002). The core model is used to classify all words whose label in the dependency representation is *not* `MOD` (case-by-assignment); whereas, the agreement model is used to classify all words

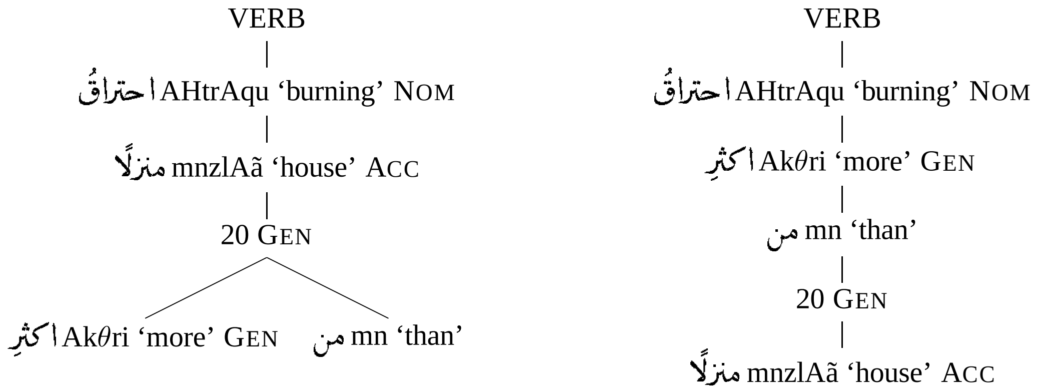

VERB
احتراقُ AHtrAqu 'burning' NOM
منزلًا mnzlAã 'house' ACC
20 GEN
اكثرِ Akθri 'more' GEN
من mn 'than'
VERB
احتراقُ AHtrAqu 'burning' NOM
اكثرِ Akθri 'more' GEN
من mn 'than'
20 GEN
منزلًا mnzlAã 'house' ACC

Figure 2: Two possible dependency trees for the phrase structure tree in Figure 1, meaning 'burning of more than 20 houses'; the tree on the left, our Basic Representation, represents a standard predicate-argument-modification style tree, while the tree on the right represents the chain of case assignment and is our Revised Representation

whose label is MOD (case-by-agreement). We handle conjunctions in the statistical system differently from the rule-based system: we resolve conjunctions so that conjoined words are labeled exactly the same. For example, in *John and Mary went to the store*, both *John* and *Mary* would have the subject label, even though *Mary* has a conjunction label in the raw dependency tree. Both models are trained only on those words which are marked for case in the treebank.

6.1 The Core Model

The core model uses the following features of a word:

- the word's POS tag;
- the conjunction of the word's POS tag and its arc label;
- the word's last length-one and length-two suffixes (to model written case morphemes);
- the conjunction of the word's arc label, its POS tag, and its parent's POS tag;
- if the word is the object of a preposition, the preposition it is the object of;
- whether the word is a PRD child of a verb (with the identity of that verb conjoined if so);
- if the word has a sister which is a subordinating conjunction, and if so, that conjunction conjoined with its arc label;
- whether the word is in an embedded clause conjoined with its arc label under the verb of the embedded clause;
- if the word is a PRD child of a verb, the verb;
- the word's left sister's POS tag conjoined with this word's arc label and its sister's arc label;
- whether the word's sister depends on the word or something else;
- and the left sister's terminal symbol.

Arabic words which do not overtly show case are still determined for purposes of resolving agreement. The classifier is applied to these cases at runtime anyway.

6.2 The Agreement Model

The agreement model uses the following features of a word:

- the word itself;
- the word's last length-one and length-two suffixes;
- and the conjunction of the word's POS tag and the case of what it agrees with.

Since words may get their case by agreement with other words which themselves get their case by agreement, the agreement model is applied repeatedly until case has been determined for all words.

System	Basic	Revised
Rule-based	93.5	94.4
Statistical	**94.0**	**95.8**

Table 1: Accuracies of various approaches on the test set in both basic and revised dependency representations.

7 Results

The performance of our two systems on the test data set is shown in table 1. There are three points to note: first, even in the basic representation, the statistical system reduces error over the rule-based system by 7.7%. Second, the revised representation helps tremendously, resulting in a 13.8% reduction in error for the rule-based system and 30% for the statistical system. Finally, the statistical system gains much more than the rule-based system from the improved representation, increasing the gap between them to a 25% reduction in error.

8 Error Analysis

We took a sample of 105 sentences (around 10%) from our development data prepared in the revised representation. Our rule-based system accuracy for the sample is about 94.1% and our statistical system accuracy is 96.2%. Table 2 classifies the different types of errors found. The first and second rows list the errors made by the statistical and rule-based systems, respectively. The third row lists errors made by the statistical system only. The fourth row lists errors made by the rule-based system only. And the fifth row lists errors made by both. The second column indicates the count of all errors. The rest of the columns specify the error types as: system errors, gold POS errors or gold tree errors. The gold POS and tree errors are treebank errors that misguide our systems. They represent 69% of all statistical system errors and 86% of all rule-based system errors. Gold POS errors represent around 35-40% of all gold errors. They most commonly include the wrong POS tag or the wrong case. One example of such errors is the mis-annotation of the ACC case to a GEN for a diptote nominal (which are indistinguishable out of context). Gold tree errors are primarily errors in the dash-tags used (or missing) in the treebank or attachment errors that are inconsistent with the gold POS tag.

The rule-based system errors involve various constructions that were not addressed in our study, e.g. flat adjectival phrases or non S constructions at the highest level in a tree (e.g. FRAG or NP). The majority of the statistical system errors involve agreement decisions and incorrect choice of case despite the presence of the dash-tags. The ratio of system errors for the statistical system is 31% (twice as much as those of the rule-based system's 14%). Thus, it seems that the statistical system manages to learn some of the erroneous noise in the treebank.

9 Discussion

9.1 Accomplishments

We have developed a system that determines case for nominals in MSA. This task is a major source of errors in full diacritization of Arabic. We use a gold-standard syntactic tree, and obtain an error rate of about 4.2%, with a machine learning based system outperforming a system using hand-written rules. A careful error analysis suggests that when we account for annotation errors in the gold standard, the error rate drops to 0.8%, with the hand-written rules outperforming the machine learning-based system.

9.2 Lessons Learned

We can draw several general conclusions from our experiments.

- The features relevant for the prediction of complex linguistic phenomena cannot necessarily be easily read off from the given representation of the data. Sometimes, due to data sparseness and/or limitations in the machine learning paradigm used, we need to extract features from the available representation in a manner that profoundly changes the representation (as is done in bilexical parsing (Collins, 1997)). Such transformations require a deep understanding of the linguistic phenomena on the part of the researchers.
- Researchers developing hand-written rules may follow an empirical methodology in natural language processing if they use data sets to develop and test the rules — the only true methodological difference between machine learning and this kind of hand-writing of rules

ERRORS	COUNT	SYSTEM	GOLD POS	GOLD TREE
All Statistical	45	14	11	20
All Rule-based	70	10	24	36
Statistical only	13	11	0	2
Rule-based only	38	7	13	18
Statistical $\bigcap$ Rule-based	32	3	11	18

Table 2: Results of Error Analysis

is the type of learning (human or machine). For certain phenomena, machine learning may result in only a small or no improvement in performance over hand-written rules.

- Error analysis remains a crucial part of any empirical work in natural language processing. Not only does it contribute insight into how the system can be improved, it also reveals problems with the underlying data. Sometimes the problems are just part of the noise in the data, but sometimes the problems can be fixed. Annotations on data are not themselves naturally occurring data and thus may be subject to critique. Note that an error analysis requires a good understanding of the linguistic phenomena and of the data.

9.3 Outlook

Our work was motivated in two ways: to help treebanking, and to develop tools for automatic case determination from unannotated text. For the first goal, our error analysis has shown that 86% of the errors found by our hand-written rules are in fact treebank errors. Furthermore, we suspect that the hand-written rules have very few false positives (i.e., cases in which the treebank has been annotated in error but our rules predict exactly that error). Thus we believe that our tool can serve an important function in improving the treebank annotation.

For our second motivation, the next step will be to adapt our feature extraction to work on the output of parsers, which typically exclude dash-tags. We note that for many contexts, we do not currently rely on dash-tags but rather identify the relevant structures on our own (such as idafa, tamyiz, and so on). We suspect that the machine learning-based approach will outperform the hand-written rules, as it can learn typical errors the parser makes. As the treebank will soon be revised and hand-checked, we will postpone this work until the new release of the treebank, which will allow us to train better parsers as the data will be more consistent.

Acknowledgements

The research presented here was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract Nos. HR0011-06-C-0023, HR0011-06-C-0022 and HR0011-06-1-0003. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

Ron Buckley. 2004. *Modern Literary Arabic: A Reference Grammar*. Librairie du Liban.

Tim Buckwalter. 2002. Buckwalter Arabic morphological analyzer version 1.0.

Michael Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the ACL (jointly with the 8th Conference of the EACL)*, pages 16–23, Madrid, Spain.

Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL07)*.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Clive Holes. 2004. *Modern Arabic: Structures, Functions, and Varieties*. Georgetown University Press. Revised Edition.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank :

Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Rani Nelken and Stuart Shieber. 2005. Arabic Diacritization Using Weighted Finite-State Transducers. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages at 43rd Meeting of the Association for Computational Linguistics (ACL'05)*, pages 79–86, Ann Arbor, Michigan.

Otakar Smrž and Jan Hajič. 2006. The Other Arabic Treebank: Prague Dependencies and Functions. In Ali Farghaly, editor, *Arabic Computational Linguistics: Current Implementations*. CSLI Publications.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Workshop on Computational Approaches to Arabic Script-based Languages*, pages 66–73, Geneva, Switzerland.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum Entropy Based Restoration of Arabic Diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584, Sydney, Australia.

Mandarin Part-of-Speech Tagging and Discriminative Reranking

Zhongqiang Huang[1]
[1]Purdue University
West Lafayette, IN 47907
zqhuang@purdue.edu

Mary P. Harper[1,2]
[2]University of Maryland
College Park, MD 20742
mharper@casl.umd.edu

Wen Wang
SRI International
Menlo Park, CA 94025
wwang@speech.sri.com

Abstract

We present in this paper methods to improve HMM-based part-of-speech (POS) tagging of Mandarin. We model the emission probability of an unknown word using all the characters in the word, and enrich the standard left-to-right trigram estimation of word emission probabilities with a right-to-left prediction of the word by making use of the current and next tags. In addition, we utilize the RankBoost-based reranking algorithm to rerank the N-best outputs of the HMM-based tagger using various n-gram, morphological, and dependency features. Two methods are proposed to improve the generalization performance of the reranking algorithm. Our reranking model achieves an accuracy of 94.68% using n-gram and morphological features on the Penn Chinese Treebank 5.2, and is able to further improve the accuracy to 95.11% with the addition of dependency features.

1 Introduction

Part-of-speech (POS) tagging is potentially helpful for many advanced natural language processing tasks, for example, named entity recognition, parsing, and sentence boundary detection. Much research has been done to improve tagging performance for a variety of languages. The state-of-the-art systems have achieved an accuracy of 97% for English on the Wall Street Journal (WSJ) corpus (which contains 4.5M words) using various models (Brants, 2000; Ratnaparkhi, 1996; Thede and Harper, 1999). Lower accuracies have been reported in the literature for Mandarin POS tagging (Tseng et al., 2005; Xue et al., 2002). This is, in part, due to the relatively small size and the different annotation guidelines (e.g., granularity of the tag set) for the annotated corpus of Mandarin. Xue at el. (2002) and Tseng at el. (2005) reported accuracies of 93% and 93.74% on CTB-I (Xue et al., 2002) (100K words) and CTB 5.0 (500K words), respectively, each using a Maximum Entropy approach. The characteristics of Mandarin make it harder to tag than English. Chinese words tend to have greater POS tag ambiguity than English. Tseng at el. (2005) reported that 29.9% of the words in CTB have more than one POS assignment compared to 19.8% of the English words in WSJ. Moreover, the morphological properties of Chinese words complicate the prediction of POS type for unknown words.

These challenges for Mandarin POS tagging suggest the need to develop more sophisticated methods. In this paper, we investigate the use of a discriminative reranking approach to increase Mandarin tagging accuracy. Reranking approaches (Charniak and Johnson, 2005; Chen et al., 2002; Collins and Koo, 2005; Ji et al., 2006; Roark et al., 2006) have been successfully applied to many NLP applications, including parsing, named entity recognition, sentence boundary detection, etc. To the best of our knowledge, reranking approaches have not been used for POS tagging, possibly due to the already high levels of accuracy for English, which leave little room for further improvement. However, the relatively poorer performance of existing methods on Mandarin POS tagging makes reranking a much more compelling technique to evaluate. In this paper, we use reranking to improve tagging performance of an HMM tagger adapted to

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1093–1102, Prague, June 2007.

Mandarin. Hidden Markov models are simple and effective, but unlike discriminative models, such as Maximum Entropy models (Ratnaparkhi, 1996) and Conditional Random Fields (John Lafferty, 2001), they have more difficulty utilizing a rich set of conditionally dependent features. This limitation can be overcome by utilizing reranking approaches, which are able to make use of the features extracted from the tagging hypotheses produced by the HMM tagger. Reranking also has advantages over MaxEnt and CRF models. It is able to use any features extracted from entire labeled sentences, including those that cannot be incorporated into MaxEnt and CRF models due to inference difficulties. In addition, reranking methods are able to utilize the information provided by N-best lists. Finally, the decoding phase of reranking is much simpler.

The rest of the paper is organized as follows. We describe the HMM tagger in Section 2. We discuss the modifications to better handle unknown words in Mandarin and to enrich the word emission probabilities through the combination of bi-directional estimations. In Section 3, we first describe the reranking algorithm and then propose two methods to improve its performance. We also describe the features that will be used for Mandarin POS reranking in Section 3. Experimental results are given in Section 4. Conclusions and future work appear in Section 5.

2 The HMM Model

2.1 Porting English Tagger to Mandarin

The HMM tagger used in this work is a second-order HMM tagger initially developed for English by Thede and Harper (1999). This state-of-the-art second-order HMM tagger uses trigram transition probability estimations, $P(t_i|t_{i-2}t_{i-1})$, and trigram emission probability estimations, $P(w_i|t_{i-1}t_i)$. Let t_1^i denote the tag sequence $t_1, \cdots, t_i$, and w_1^i denote the word sequence $w_1, \cdots, w_i$. The tagging problem can be formally defined as finding the best tag sequence $\tau(w_1^N)$ for the word sequence w_1^N of length N as follows[1]:

$$\begin{aligned}
\tau(w_1^N) &= \arg\max_{t_1^N} P(t_1^N|w_1^N) = \arg\max_{t_1^N} \frac{P(t_1^N w_1^N)}{P(w_1^N)} \\
&= \arg\max_{t_1^N} P(t_1^N w_1^N) \qquad (1) \\
&= \arg\max_{t_1^N} \prod_i P(t_i|t_1^{i-1}w_1^{i-1})P(w_i|t_1^i w_1^{i-1}) \\
&\approx \arg\max_{t_1^N} \prod_i P(t_i|t_{i-2}t_{i-1})P(w_i|t_{i-1}t_i) \qquad (2)
\end{aligned}$$

The best tag sequence $\tau(w_1^N)$ can be determined efficiently using the Viterbi algorithm.

For estimating emission probabilities of unknown words (i.e., words that do not appear in the training data) in English (and similarly for other inflected languages), a weighted sum of $P(s_i^k|t_{i-1}t_i)$ (with k up to four) was used as an approximation, where s_i^k is the suffix of length k of word w_i (e.g., s_i^1 is the last character of word w_i). The suffix information and three binary features (i.e., whether the word is capitalized, whether the word is hyphenated, and whether the word contains numbers) are combined to estimate the emission probabilities of unknown words.

The interpolation weights for smoothing transition, emission, and suffix probabilities were estimated using the log-based Thede smoothing method (Thede and Harper, 1999) as follows:

$$\begin{aligned}
&P_{Thede}(\text{n-gram}) \\
&\quad = \lambda(\text{n-gram})P_{ML}(\text{n-gram}) + \\
&\qquad (1 - \lambda(\text{n-gram}))P_{Thede}((\text{n-1})\text{-gram})
\end{aligned}$$

where:

$$\begin{aligned}
P_{ML}(\text{n-gram}) &= \text{the ML estimation} \\
\lambda(\text{n-gram}) &= f(\text{n-gram count}) \\
f(x) &= \frac{\log_a(x+1) + b}{\log_a(x+1) + (b+1)}
\end{aligned}$$

While porting the HMM-based English POS tagger to Mandarin is fairly straightforward for words seen in the training data, some thought is required to handle unknown words due to the morphology differences between the two languages. First, in Mandarin, there is no capitalization and no hyphenation. Second, although Chinese has morphology, it is not the same as in English; words tend to contain far fewer characters than inflected words in English, so word endings will tend to be short, say one or two characters long. Hence, in our baseline model (denoted *HMM baseline*), we simply utilize word endings of up to two characters in length along with a binary feature of whether the word contains numbers or not. In the next two subsections, we describe two ways in which we enhance this simple HMM baseline model.

[1] We assume that symbols exist implicitly for boundary conditions.

2.2 Improving the Mandarin Unknown Word Model

Chinese words are quite different from English words, and the word formation process for Chinese words can be quite complex (Packard, 2000). Indeed, the last characters in a Chinese word are, in some cases, most informative of the POS type, while for others, it is the characters at the beginning. Furthermore, it is not uncommon for a character in the middle of a word to provide some evidence for the POS type of the word. Hence, we chose to employ a rather simple but effective method to estimate the emission probability, $P(w_i|t_{i-1}, t_i)$, of an unknown word, w_i. We use the geometric average[2] of the emission probability of the characters in the word, i.e., $P(c_k|t_{i-1}, t_i)$ with c_k being the k-th character in the word. Since some of the characters in w_i may not have appeared in any word tagged as t_i in that context in the training data, only characters that are observed in this context are used in the computation of the geometric average, as shown below:

$$P(w_i|t_{i-1}, t_i) = \sqrt[n]{\prod_{c_k \in w_i, P(c_k|t_{i-1}, t_i) \neq 0} P(c_k|t_{i-1}, t_i)} \quad (3)$$

where

$$n = |\{c_k \in w_i | P(c_k|t_{i-1}, t_i) \neq 0\}|$$

2.3 Bi-directional Word Probability Estimation

In Equation 2, the word emission probability $P(w_i|t_{i-1}t_i)$ is a left-to-right prediction that depends on the current tag t_i associated with w_i, as well as its previous tag t_{i-1}. Although the interaction between w_i and the next tag t_{i+1} is captured to some extent when t_{i+1} is generated by the model, this implicit interaction may not be as effective as adding the information more directly to the model. Hence, we chose to apply the constraint explicitly in our HMM framework by replacing $P(w_i|t_{i-1}t_i)$ in Equation 2 with $P^{\lambda}(w_i|t_{i-1}t_i)P^{1-\lambda}(w_i|t_i t_{i+1})$ for both known and unknown words, with $\tau(w_1^N)$ determined by:

$$\begin{aligned} \tau(w_1^N) &= \arg\max_{t_1^N} \prod_i (P(t_i|t_{i-2}t_{i-1}) \times \\ &\qquad P^{\lambda}(w_i|t_{i-1}t_i)P^{1-\lambda}(w_i|t_i t_{i+1})) \end{aligned} \quad (4)$$

This corresponds to a mixture model of two generation paths, one from the left and one from the right, to approximate $\tau(w_1^N)$ in Equation 1 in a different way.

$$\begin{aligned} \tau(w_1^N) &= \arg\max_{t_1^N} P(t_1^N w_1^N) \\ &= \arg\max_{t_1^N} P(t_1^N)P(w_1^N|t_1^N) \\ P(t_1^N) &\approx \prod_i P(t_i|t_{i-1}t_{i-2}) \\ P(w_1^N|t_1^N) &= P^{\lambda}(w_1^N|t_1^N)P^{1-\lambda}(w_1^N|t_1^N) \\ &\approx \prod_i P^{\lambda}(w_i|t_{i-1}t_i)P^{1-\lambda}(w_i|t_i t_{i+1}) \end{aligned}$$

In this case, the decoding process involves the computation of three local probabilities, i.e., $P(t_i|t_{i-2}t_{i-1})$, $P(w_i|t_{i-1}t_i)$, and $P(w_i|t_i t_{i+1})$. By using a simple manipulation that shifts the time index of $P(w_i|t_i t_{i+1})$ in Equation 4 by two time slices[3] (i.e., by replacing $P(w_i|t_i t_{i+1})$ with $P(w_{i-2}|t_{i-2}t_{i-1})$), we are able to compute $\tau(w_1^N)$ in Equation 4 with the same asymptotic time complexity of decoding as in Equation 2.

3 Discriminative Reranking

In this section, we describe our use of the RankBoost-based (Freund and Schapire, 1997; Freund et al., 1998) discriminative reranking approach that was originally developed by Collins and Koo (2005) for parsing. It provides an additional avenue for improving tagging accuracy, and also allows us to investigate the impact of various features on Mandarin tagging performance. The reranking algorithm takes as input a list of candidates produced by some probabilistic model, in our case the HMM tagger, and reranks these candidates based on a set of features. We first introduce Collins' reranking algorithm in Subsection 3.1, and then describe two modifications in Subsections 3.2 and 3.3 that were designed to improve the generalization performance of the reranking algorithm for our POS tagging task. The reranking features that are used for POS tagging are then described in Subsection 3.4.

3.1 Collins' Reranking Algorithm

For training the reranker for the POS tagging task, there are n sentences $\{s_i : i = 1, \cdots, n\}$ each with n_i candidates $\{x_{i,j} : j = 1, \cdots, n_i\}$ along with

[2] Based on preliminary testing, the geometric average provided greater tag accuracy than the arithmetic average.

[3] Replacing $P(w_i|t_i t_{i+1})$ with $P(w_{i-1}|t_{i-1}t_i)$ also gives the same solution.

the log-probability $L(x_{i,j})$ produced by the HMM tagger. Each tagging candidate $x_{i,j}$ in the training data has a "goodness" score $Score(x_{i,j})$ that measures the similarity between the candidate and the gold reference. For tagging, we use tag accuracy as the similarity measure. Without loss of generality, we assume that $x_{i,1}$ has the highest score, i.e., $Score(x_{i,1}) \geq Score(x_{i,j})$ for $j = 2, \cdots, n_i$. To summarize, the training data consists of a set of examples $\{x_{i,j} : i = 1, \cdots, n; j = 1, \cdots, n_i\}$, each along with a "goodness" score $Score(x_{i,j})$ and a log-probability $L(x_{i,j})$.

A set of indicator functions $\{h_k : k = 1, \cdots, m\}$ are used to extract binary features $\{h_k(x_{i,j}) : k = 1, \cdots, m\}$ on each example $x_{i,j}$. An example of an indicator function for POS tagging is given below:

$$\begin{array}{rcl} h_{2143}(x) & = & 1 \text{ if } x \text{ contains n-gram ``go/VV to''} \\ & & 0 \text{ otherwise} \end{array}$$

Each indicator function h_k is associated with a weight parameter α_k which is real valued. In addition, a weight parameter α_0 is associated with the log-probability $L(x_{i,j})$. The ranking function of candidate $x_{i,j}$ is defined as $\alpha_0 L(x_{i,j}) + \sum\limits_{k=1}^{m} \alpha_k h_k(x_{i,j})$.

The objective of the training process is to set the parameters $\bar{\alpha} = \{\alpha_0, \alpha_1, \cdots, \alpha_m\}$ to minimize the following loss function Loss$(\bar{\alpha})$ (which is an upper bound on the training error):

$$\text{Loss}(\bar{\alpha}) = \sum_i \sum_{j=2}^{n_i} S_{i,j} e^{-M_{i,j}(\bar{\alpha})}$$

where $S_{i,j}$ is the weight function that gives the importance of each example, and $M_{i,j}(\bar{\alpha})$ is the margin:

$$\begin{array}{rcl} S_{i,j} & = & Score(x_{i,1}) - Score(x_{i,j}) \\ M_{i,j}(\bar{\alpha}) & = & \alpha_0(L(x_{i,1}) - L(x_{i,j})) + \\ & & \displaystyle\sum_{k=1}^{m} \alpha_k(h_k(x_{i,1}) - h_k(x_{i,j})) \end{array}$$

All of the α_i's are initially set to zero. The value of α_0 is determined first to minimize the loss function and is kept fixed afterwards. Then a greedy sequential [4] optimization method is used in each iteration (i.e., a boosting round) to select the feature that has the most impact on reducing the loss function and then update its weight parameter accordingly. For each $k \in \{1, \cdots, m\}$, $(h_k(x_{i,1}) - h_k(x_{i,j}))$ can only take one of the three values: +1, -1, or 0. Thus the training examples can be divided into three subsets with respect to k:

$$\begin{array}{rcl} A_k^+ & = & \{(i,j) : (h_k(x_{i,1}) - h_k(x_{i,j})) = +1\} \\ A_k^- & = & \{(i,j) : (h_k(x_{i,1}) - h_k(x_{i,j})) = -1\} \\ A_k^0 & = & \{(i,j) : (h_k(x_{i,1}) - h_k(x_{i,j})) = 0\} \end{array}$$

The new loss after adding the update parameter δ to the parameter α_k is shown below:

$$\begin{array}{rcl} \text{Loss}(\bar{\alpha}, k, \delta) & = & \displaystyle\sum_{(i,j)\in A_k^+} S_{i,j} e^{-M_{i,j}(\bar{\alpha})-\delta} + \\ & & \displaystyle\sum_{(i,j)\in A_k^-} S_{i,j} e^{-M_{i,j}(\bar{\alpha})+\delta} + \\ & & \displaystyle\sum_{(i,j)\in A_k^0} S_{i,j} e^{-M_{i,j}(\bar{\alpha})} \\ & = & e^{-\delta} W_k^+ + e^{\delta} W_k^- + W_k^0 \end{array}$$

The best feature/update pair (k^*, δ^*) that minimizes Loss$(\bar{\alpha}, k, \delta)$ is determined using the following formulas:

$$k^* = \arg\max_k \left| \sqrt{W_k^+} - \sqrt{W_k^-} \right| \quad (5)$$

$$\delta^* = \frac{1}{2} \log \frac{W_{k^*}^+}{W_{k^*}^-} \quad (6)$$

The update formula in Equation 6 is problematic when either $W_{k^*}^+$ or $W_{k^*}^-$ is zero. W_k^+ is zero if h_k never takes on a value 1 for any $x_{i,1}$ with value 0 on a corresponding $x_{i,j}$ for $j = 2, \cdots, n_i$ (and similarly for W_k^-). Collins introduced a smoothing parameter ϵ to address this problem, resulting in a slight modification to the update formula:

$$\delta^* = \frac{1}{2} \log \frac{W_{k^*}^+ + \epsilon Z}{W_{k^*}^- + \epsilon Z} \quad (7)$$

The value of ϵ plays an important role in this formula. If ϵ is set too small, the smoothing factor ϵZ would not prevent setting δ^* to a potentially overly large absolute value, resulting in over-fitting. If ϵ is set too large, then the opposite condition of undertraining could result. The value of ϵ is determined based on a development set.

[4] Parallel optimization algorithms exist and have comparable performance according to (Collins et al., 2002).

3.2 Update Once

Collins' method allows multiple updates to the weight of a feature based on Equations 5 and 7. We found that for those features for which either W_k^+ or W_k^- equals zero, the update formula in Equation 7 can only increase their weight (in absolute value) in one direction. Although these features are strong and useful, setting their weights too large can be undesirable in that it limits the use of other features for reducing the loss.

Based on this analysis, we have developed and evaluated an *update-once* method, in which we use the update formula in Equation 7 but limit weight updates so that once a feature is selected on a certain iteration and its weight parameter is updated, it cannot be updated again. Using this method, the weights of the strong features are not allowed to prevent additional features from being considered during the training phase.

3.3 Regularized Reranking

Although the update-once method may attenuate over-fitting to some extent, it also prevents adjusting the value of any weight parameter that is initially set too high or too low in an earlier boosting round. In order to design a more sophisticated weight update method that allows multiple updates in both directions while penalizing overly large weights, we have also investigated the addition of a regularization term $R(\bar{\alpha})$, an exponential function of $\bar{\alpha}$, to the loss function:

$$\begin{aligned}\text{RegLoss}(\bar{\alpha}) &= \sum_i \sum_{j=2}^{n_i} S_{i,j} e^{-M_{i,j}(\bar{\alpha})} + R(\bar{\alpha}) \\ R(\bar{\alpha}) &= \sum_{k=1}^{m} p_k \cdot (e^{-\alpha_k} + e^{\alpha_k} - 2)\end{aligned}$$

where p_k is the penalty weight of parameter α_k. The reason that we chose this form of regularization is that $(e^{-\alpha_k} + e^{\alpha_k} - 2)$ is a symmetric, monotonically decreasing function of $|\alpha_k|$, and more importantly it provides a closed analytical expression of the weight update formula similar to Equations 5 and 6. Hence, the best feature/update pair for the regularized loss function is defined as follows:

$$\begin{aligned}k^* &= \arg\max_k \left| \sqrt{W_k^+ + p_k e^{-\alpha_k}} - \sqrt{W_k^- + p_k e^{+\alpha_k}} \right| \\ \delta^* &= \frac{1}{2} \log \frac{W_{k^*}^+ + p_{k^*} e^{-\alpha_{k^*}}}{W_{k^*}^- + p_{k^*} e^{+\alpha_{k^*}}}\end{aligned}$$

There are many ways of choosing p_k, the penalty weight of α_k. In this paper, we use the values of $\beta \cdot (W_k^+ + W_k^-)$ at the beginning of the first iteration (after α_0 is determined) for p_k, where β is a weighting parameter to be tuned on the development set. The regularized weight update formula has many advantages. It is always well defined no matter what value W_k^+ and W_k^- take, in contrast to Equation 6. For all features, even in the case when either W_k^+ or W_k^- equals zero, the regularized update formula allows weight updates in two directions. If the weight is small, W_k^+ and W_k^- have more impact on determining the weight update direction, however, when the weight becomes large, the regularization factors $p_k e^{-\alpha}$ and $p_k e^{+\alpha}$ favor reducing the weight.

3.4 Reranking Features

A reranking model has the flexibility of incorporating any type of feature extracted from N-best candidates. For the work presented in this paper, we examine three types of features. For each window of three word/tag pairs, we extract all the n-grams, except those that are comprised of only one word/tag pair, or only tags, or only words, or do not include either the word or tag in the center word/tag pair. These constitute the *n-gram* feature set.

In order to better handle unknown words, we also extract the two most important types of *morphological* features[5] that were utilized in (Tseng et al., 2005) for those words that appear no more than seven times (following their convention) in the training set:

Affixation features: we use character n-gram prefixes and suffixes for n up to 4. For example, for word/tag pair 资料袋/NN (Information-Bag, i.e., folder), we add the following features: (prefix1, 资, NN), (prefix2, 资料, NN), (prefix3, 资料袋, NN), (suffix1, 袋, NN), (suffix2, 料袋, NN), (suffix3, 资料袋, NN).

AffixPOS features[6]: we used the training set to build a prefix/POS and suffix/POS dictionary associating possible tags with each prefix and

[5] Tseng at el. also used other morphological features that require additional resources to which we do not have access.

[6] AffixPOS features are somewhat different from the CTB-Morph features used in (Tseng et al., 2005), where a morpheme/POS dictionary with the possible tags for all morphemes in the training set was used instead of two separate dictionaries for prefix and suffix. AffixPOS features perform slightly better in our task than the CTB-morph features.

suffix in the training set. The AffixPOS features indicate the set of tags a given affix could have. For the same example 资料袋/NN, 资 occurred as prefix in both NN and VV words in the training data. So we add the following features based on the prefix 资: (prefix, 资, NN, 1, NN), (prefix, 资, VV, 1, NN), and (prefix, 资, X, 0, NN) for every tag X not in {NN, VV}, where 1 and 0 are indicator values. Features are extracted in the similar way for the suffix 袋.

The n-gram and morphological features are easy to compute, however, they have difficulty in capturing the long distance information related to syntactic relationships that might help POS tagging accuracy. In order to examine the effectiveness of utilizing syntactic information in tagging, we have also experimented with *dependency* features that are extracted based on automatic parse trees. First a bracketing parser (the Charniak parser (Charniak, 2000) in our case) is used to generate the parse tree of a sentence, then the *const2dep* tool developed by Hwa was utilized to convert the bracketing tree to a dependency tree based on the head percolation table developed by the second author. The dependency tree is comprised of a set of dependency relations among word pairs. A dependency relation is a triple ⟨*word-a*, *word-b*, *relation*⟩, in which *word-a* is governed by *word-b* with grammatical relation denoted as *relation*. For example, in the sentence "西藏(Tibet) 经济(economy) 建设(construction) 取得(achieves) 显著(significant) 成绩(accomplishments)", one example dependency relation is ⟨取得, 成绩, *mod*⟩. Given these dependency relations, we then extract dependency features (in total 36 features for each relation) by examining the POS tags of the words for each tagging candidate of a sentence. The relative positions of the word pairs are also taken into account for some features. For example, if 取得 and 成绩 in the above sentence are tagged as VV and NN respectively in one candidate, then two example dependency features are (dep-1, 取得, VV, 成绩, NN, *mod*), (dep-14, 取得, VV, NN, *right*, *mod*), in which dep-1 and dep-14 are feature types and *right* indicates that *word-b* (取得) is to the right of *word-a* (成绩).

4 Experiments

4.1 Data

The most recently released Penn Chinese Treebank 5.2 (denoted CTB, released by LDC) is used in our experiments. It contains 500K words, 800K characters, 18K sentences, and 900 data files, including articles from the Xinhua news agency (China-Mainland), Information Services Department of HKSAR (Hongkong), and Sinorama magazine (Taiwan). Its format is similar to the English WSJ Penn Treebank, and it was carefully annotated. There are 33 POS tags used, to which we add tags to discriminate among punctuation types. The original POS tag for punctuation was PU; we created new POS tags for each distinct punctuation type (e.g., PU-?).

The CTB corpus was collected during different time periods from different sources with a diversity of articles. In order to obtain a representative split of training, development, and test sets, we divide the whole corpus into blocks of 10 files by sorted order. For each block, the first file is used for development, the second file is used for test, and the remaining 8 files are used for training. Table 1 gives the basic statistics on the data. The development set is used to determine the parameter λ in Equation 4, the smoothing parameter ϵ in Equation 7, the weight parameter β described in Section 3.3, and the number of boosting rounds in the reranking model. In order to train the reranking model, the method in (Collins and Koo, 2005) is used to prepare the N-best training examples. We divided the training set into 20 chunks, with each chunk N-best tagged by the HMM model trained on the combination of the other 19 chunks. The development set is N-best tagged by the HMM model trained on the training set, and the test set is N-best tagged by the HMM model trained on the combination of the training set and the development set.

	Train	Dev	Test
#Sentences	14925	1904	1975
#Words	404844	51243	52900

Table 1: The basic statistics on the data.

In the following subsections, we will first examine the HMM models alone to determine the best HMM configuration to use to generate the N-best candidates, and then evaluate the reranking models. Finally, we compare our performance with previous work. In this paper, we use the sign test with $p \leq 0.01$ to evaluate the statistical significance of the difference between the performances of two models.

4.2 Results of the HMM taggers

The baseline HMM model ported directly from the English tagger, as described in Subsection 2.1, has an overall tag accuracy of 93.12% on the test set, which is fairly low compared to the 97% accuracy of many state-of-the-art taggers on WSJ for English.

By approximating the unknown word emission probability using the characters in the word as in Equation 3, the performance of the HMM tagger improves significantly to 93.43%, suggesting that characters in different positions of a Chinese word help to disambiguate the word class of the entire word, in contrast to English for which suffixes are most helpful.

Figure 1 depicts the impact of combining the left-to-right and right-to-left word emission models using different weighting values (i.e., λ) on the development set. Note that emission probabilities of unknown words are estimated based on characters using the same λ for combination. When $\lambda = 1.0$, the model uses only the standard left-to-right prediction of words, while when $\lambda = 0$ it uses only the right-to-left estimation. It is interesting to note that the right-to-left estimation results in greater accuracy than the left-to-right estimation. This might be because there is stronger interaction between a word and its next tag. Also as shown in Figure 1, the estimations in the two directions are complementary to each other, with $\lambda = 0.5$ performing best. The performance of the HMM taggers on the test set is given in Table 2 for the best operating point, as well as the two other extreme operating points to compare the left-to-right and right-to-left constraints. Our best HMM tagger further improves the tag accuracy significantly from 93.43% ($\lambda = 1.0$) to 94.01% ($\lambda = 0.5$).

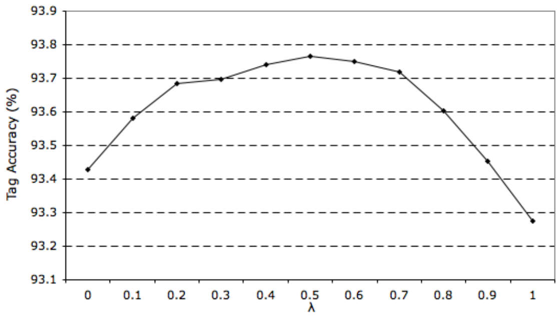


Figure 1: The accuracy of the HMM tagger on the development set with various λ values for combining the word emission probabilities.

	Overall	Known	Unknown
HMM baseline	93.12%	94.65%	69.08%
HMM, λ=1.0	93.43%	94.71%	73.41%
HMM, λ=0.0	93.65%	94.88%	74.23%
HMM, λ=0.5	94.01%	95.21%	75.15%

Table 2: The performance of various HMM taggers on the test set.

4.3 Results of the Reranking Models

The HMM tagger with the best accuracy (i.e., the one with $\lambda = 0.5$ in Table 2) is used to generate the N-Best tagging candidates, with a maximum of 100 candidates. As shown in Table 3, a maximum of 100-Best provides a reasonable margin for improvement in the reranking task.

We first test the performance of the reranking methods using only the n-gram feature set, which contains around 18 million features. Later, we will investigate the addition of morphological features and dependency features. The smoothing parameter ϵ (for Collins' method and the update-once method) and the weight parameter β (for the regularization method) both have great impact on reranking performance. We trained various reranking models with ϵ values of $0.0001 \times \{1, 2.5, 5, 7.5, 10, 25, 50, 75, 100\}$, and β values of $\{0.1, 0.25, 0.5, 0.75, 1\}$. For all these parameter values, 600,000 rounds of iterations were executed on the training set. The development set was used to determine the early stopping point in training. If not mentioned explicitly, all the results reported are based on the best parameters tuned on the development set.

	1-Best	50-Best	100-Best
train	93.48%	96.96%	97.13%
dev	93.75%	97.68%	97.84%
test	93.19%	97.19%	97.35%

Table 3: The oracle tag accuracies of the 1-Best, 50-Best, and 100-Best candidates in the training, development, and test sets for the reranking experiments. Note that the tagging candidates are prepared using the method described in Subsection 4.1.

Table 4 reports the performance of the best HMM tagger and the three reranking taggers on the test set. All three reranking methods improve the HMM tagger significantly. Also, the update-once and regularization methods both outperform Collins' original training method significantly.

	Overall	Known	Unknown
HMM, λ=0.5	94.01%	95.21%	75.15%
Collins	94.38%	95.56%	75.85%
Update-once	94.50%	95.67%	76.13%
Regularized	94.54%	95.70%	76.48%

Table 4: The performance on the test set of the HMM tagger, and the reranking methods using the n-gram features.

We observed that no matter which value the smoothing parameter ϵ takes, there are only about 10,000 non-zero features finally selected by Collins' original method. In contrast, the two new methods select substantially more features, as shown in Table 5. As mentioned before, there are some strong features that only appear in positive or negative samples, i.e., either W_k^+ or W_k^- equals zero. Although introducing the smoothing parameter ϵ in Equation 7 prevents infinite weight values, the update to the feature weights is no longer optimal (in terms of minimizing the error function). Since the update is not optimal, subsequent iterations may still focus on these features (and thus ignore other weaker but informative features) and always increase their weights in one direction, leading to biased training.

The update-once method at each iteration selects a new feature that has the most impact in reducing the training loss function. It has the advantage of preventing increasingly large weights from being assigned to the strong features, enabling the update of other features. The regularization method allows multiple updates and also penalizes large weights. Once a feature is selected and has its weight updated, no matter how strong the feature is, the weight value is optimal in terms of the current weights of other features, so that the training algorithm would choose another feature to update. A previously selected feature may be selected again if it becomes suboptimal due to a change in the weights of other features.

	#iterations	#features	percent
Collins	115400	10020	8.68%
Update-once	545100	545100	100%
Regularized	92500	70131	75.82%

Table 5: The number of iterations (for the best performance), the number of selected features, and the percentage of selected features, by Collins' method, the update-once method, and the regularization method on the development set.

	Overall	Known	Unknown
HMM, λ=0.5	94.01%	95.21%	75.15%
Collins	94.44%	95.55%	77.05%
Update-once	94.68%	95.68%	78.91%
Regularized	94.64%	95.71%	77.84%

Table 6: The performance on the test set of the HMM tagger and the reranking methods using n-gram and morphological features.

We next add morphological features to the n-gram features selected by the reranking methods[7]. As can be seen by comparing Table 6 to Table 4, morphological features improve the tagging accuracy of unknown words. It should be noted that the improvement made by both update-one and regularization methods is statistically significant over using n-gram features alone; however, the improvement by Collins' original method is not significant. This suggests that the two new methods are able to utilize a greater variety of features than the original method.

We trained several Charniak parsers using the same method for the HMM taggers to generate automatic parse trees for training, development, and test data. The update-once method is used to evaluate the effectiveness of dependency features for reranking, as shown in Table 7. The parser has an overall tagging accuracy that is greater than that of the best HMM tagger, but worse than that of the reranking models using n-gram and morphological features. It is interesting to note that reranking with the dependency features alone improves the tagging accuracy significantly, outperforming reranking models using n-gram and morphological features. This suggests that the long distance features based on the syntactic structure of the sentence are very beneficial for POS tagging of Mandarin. Moreover, n-gram and morphological features are complementary to the dependency features, with their combination performing the best. The n-gram features improve the accuracy on known words, while the morphological features improve the accuracy on unknown words. The best accuracy of 95.11% is an 18% relative reduction in error compared to the best HMM tagger.

[7] Because the size of the combined feature set of all n-gram features and morphological features is too large to be handled by our server, we chose to add morphological features to the n-gram features selected by the reranking methods, and then retrain the reranking model.

	Overall	Known	Unknown
Parser	94.31%	95.57%	74.52%
dep	94.93%	96.01%	77.87%
dep+ngram	95.00%	96.11%	77.49%
dep+morph	94.98%	96.01%	78.79%
dep+ngram+morph	95.11%	96.12%	79.32%

Table 7: The tagging performance of the parser and the update-once reranking models with dependency features and their combination with n-gram and morphological features.

4.4 Comparison to Previous Work

So how is our performance compared to previous work? When working on the same training/test data (CTB5.0 with the same pre-processing procedures) as in (Tseng et al., 2005), our HMM model obtained an accuracy of 93.72%, as compared to their 93.74% accuracy. Our reranking model[8] using n-gram and morphological features improves the accuracy to 94.16%. Note that we did not use all the morphological features as in (Tseng et al., 2005), which would probably provide additional improvement. The dependency features are expected to further improve the performance, although they are not included here in order to provide a relatively fair comparison.

5 Conclusions and Future Work

We have shown that the characters in a word are informative of the POS type of the entire word in Mandarin, reflecting the fact that the individual Chinese characters carry POS information to some degree. The syntactic relationship among characters may provide further information, which we leave as future work. We have also shown that the additional right-to-left estimation of word emission probabilities is useful for HMM tagging of Mandarin. This suggests that explicit modeling of bidirectional interactions captures more sequential information. This could possibly help in other sequential modeling tasks.

We have also investigated using the reranking algorithm in (Collins and Koo, 2005) for the Mandarin POS tagging task, and found it quite effective in improving tagging accuracy. The original algorithm has a tendency to focus on a small subset of strong features and ignore some of the other useful features. We were able to improve the performance of the reranking algorithm by utilizing two different methods that make better use of more features. Both are simple and yet effective. The effectiveness of dependency features suggests that syntax-based long distance features are important for improving part-of-speech tagging performance in Mandarin. Although parsing is computationally more demanding than tagging, we hope to identify related features that can be extracted more efficiently.

In future efforts, we plan to extract additional reranking features utilizing more explicitly the characteristics of Mandarin. We also plan to extend our work to speech transcripts for Broadcast News and Broadcast Conversation corpora, and explore semi-supervised training methods for reranking.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. We gratefully acknowledge the comments from the anonymous reviewers.

References

Thorsten Brants. 2000. TnT a statistical part-of-speech tagger. In *ANLP*, pages 224–231.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 132–139, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

John Chen, Srinivas Bangalore, Michael Collins, and Owen Rambow. 2002. Reranking an n-gram supertagger. In *the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks*.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

[8] Tseng at el.'s training/test split uses up the entire CTB corpus, leaving no development data for tuning parameters. In order to roughly measure reranking performance, we use the update-once method to train the reranking model for 600,000 rounds with the other parameters tuned in Section 4. This sacrifices performance to some extent.

Michael Collins, Robert E. Schapire, and Yoram Singer. 2002. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1):253–285.

Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1(55):119–139.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 1998. An efficient boosting algorithm for combining preferences. In *the Fifteenth International Conference on Machine Learning*.

Heng Ji, Cynthia Rudin, and Ralph Grishman. 2006. Reranking algorithms for name tagging. In *HLT/NAACL 06 Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*.

Fernando Pereira John Lafferty, Andrew McCallum. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Jerome Packard. 2000. *The Morphology of Chinese*. Cambridge University Press.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*.

Brian Roark, Yang Liu, Mary Harper, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bonnie Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. Reranking for sentence boundary detection in conversational speech. In *ICASSP*.

Scott M. Thede and Mary P. Harper. 1999. A second-order hidden Markov model for part-of-speech tagging. In *ACL*, pages 175–182.

Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. 2005. Morphological features help pos tagging of unknown words across language varieties. In *the Fourth SIGHAN Workshop on Chinese Language Processing*.

Nianwen Xue, Fu dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *COLING*.

Building Domain-Specific Taggers without Annotated (Domain) Data

John E. Miller[1] Manabu Torii[2] K. Vijay-Shanker[1]

[1]Computer & Information Sciences
University of Delaware
Newark, DE 19716
{jmiller,vijay}@cis.udel.edu

[2]Biostatistics, Bioinformatics and Biomathematics
Georgetown University Medical Center
Washington, DC 20057
mt352@georgetown.edu

Abstract

Part of speech tagging is a fundamental component in many NLP systems. When taggers developed in one domain are used in another domain, the performance can degrade considerably. We present a method for developing taggers for new domains without requiring POS annotated text in the new domain. Our method involves using raw domain text and identifying related words to form a domain specific lexicon. This lexicon provides the initial lexical probabilities for EM training of an HMM model. We evaluate the method by applying it in the Biology domain and show that we achieve results that are comparable with some taggers developed for this domain.

1 Introduction

As Natural Language Processing (NLP) technology advances and more text becomes available, it is being applied more and often in specialized domains. Part of Speech (POS) tagging is often a fundamental component to these NLP applications and hence its accuracy can have a significant impact on the application's success. The success that the taggers have attained is often not replicated when the domain is changed. Degradation of accuracy in a new domain can be overcome by developing an annotated corpus for that specific domain, e.g., as in the Biology domain. However, this solution is feasible only if there is sufficient interest in the use of NLP technology in that domain, and there are sufficient funding and resources. In contrast, our approach is to use existing resources, and rapidly develop taggers for new domains without using the time and effort to develop annotated data.

In this work, we use the Wall Street Journal (WSJ) corpus (Marcus et al, 1993) and large amounts of domain-specific *raw* text to develop taggers. We evaluate our methodology in the Biology domain and show the resulting performance is competitive with some taggers built with supervised learning for that domain. Also, we note that the accuracy of taggers trained on the WSJ corpus drops off considerably when applied to this domain. Smith et al. (2005) report that the Brill tagger (1995) has an accuracy of 86.8% on 1000 sentences taken from Medline, and that the Xerox tagger (Cutting et al .1992) has an accuracy of 93.1% on the same sentences. They attribute this drop off to the fact that only 57.8% of the 10,000 most frequent words can be found in WSJ corpus. This observation provides further impetus to developing lexicon for taggers in the new domains.

In the next section, we discuss our general approach. The details of the EM training of the HMM tagger are given in Section 3. Section 4 provides details of how a domain specific lexicon is created. Next, we discuss the evaluation of our models and analysis based on the results. Section 6 discusses related work and those works from which we have taken some ideas. Section 7 has some concluding remarks.

2 Basic Methodology

Inadequate treatment of domain-specific vocabulary is often the primary cause in the degradation of performance when a tagger trained in one genre of text is ported to a new domain. The significance of out-of-vocabulary words has been noted in reduced accuracy of NLP components in the Biology

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1103–1111, Prague, June 2007. ©2007 Association for Computational Linguistics

domain (e.g., Lease and Charniak, 2005; Smith et al. 2004). The handling of domain-specific vocabulary is the focus of our approach.

It is quite common to use suffix information in the prediction of POS tags for occurrences of new words. However, its effectiveness may be limited in English, which is not a highly inflected language. However, even for English, we find that not only can suffix information be used online during tagging, but also the presence or absence of morphologically related words can provide considerable information to pre-build a lexicon that associates possible tags with words.

Consider the example of the word "broaden". While the suffix "en" may be utilized to predict the likelihood of verbal tags (VB and VBP) for the word during tagging, if we were to build a lexicon offline, the existence of the words "broadened", "broadening", "broadens" and "broad" give further evidence to treat "broaden" as a verb. This type of information has been used before in (Cucerzan and Yarowsky, 2000).

In the above example, the presence or absence of words with the suffix morphemes suggests POS tag information in two ways: 1) The presence of a suffix morpheme in a word suggests a POS tag or a small set of POS tags for the word. This is the type of information most taggers use to predict tags for unknown words during the tagging process; 2) The presence of the morpheme can also indicate possible tags for the words it attaches to. For example, the derivational morpheme "ment" indicates "government" is likely to be an NN and also that the word it attaches to, "govern" is likely to be a verb. Inflectional and derivational morphemes don't attach to words of just any POS category; they are particular. Thus, we can propose the possibility of JJ (adjective) to "broad" and VB or VBP to "govern" (based on the fact the derivational morphemes "en" and "ment" attach to them) even though by themselves they don't have any suffix information that might be indicative of JJ and VB or VBP.

Additional suffixes (that may or may not be taken from a standard list of English inflectional and derivational morphemes) can also be used. As an example, the suffix "ate" can be associated with a small set of tags: VB or VBP ("educate", "create"), JJ ("adequate", "appropriate"), and NN ("candidate", "climate"). Note the possibility or impossibility of the addition of "tion" and "ly" can help distinguish between the verbal and adjectival situations. In contrast, most taggers that use just suffix information during the tagging process will need strong contextual information (i.e., tags of nearby words) in making their prediction for each occurrence, as such suffixes can be associated with multiple tags.

To utilize such information, we need a dictionary of words in the domain for which we are interested in building a tagger. Such a dictionary will allow us to propose possible tags for a domain word such as "phosphorylate". If we can verify whether words like "phosphorylation", "phosphorylates", and "phosphorylately," are available in the domain then we can obtain considerable information regarding the possible tags that can be associated with "phosphorylate". But we cannot assume the availability of a dictionary of words in the domain. However, it would suffice to have a large text corpus, which we call ***Text-Lex***. We use it as a proxy for a domain dictionary by obtaining a list of words and their relative frequency of appearance in the domain.

Rather than using manually developed rules that assign possible tags for words based on the presence or absence of related words, we wish to apply a more empirical methodology. Since this sort of information is specific to a language rather than a domain, we can use an annotated corpus in another domain to provide exemplars. We use the WSJ (Marcus et al. 1993) corpus, a POS annotated corpus, for this purpose. For example, we can see that "phosphorylate" in the Biology domain and "create" in the WSJ corpus are similar in the sense both take on "tion", "ed", and "ing" suffixes but not "ly" for instance. Since the WSJ corpus would provide POS tag information for "create", we can use it to inform us for "phosphorylate".

The above method forms the basis for our determination of the set of tags that are to be associated with the domain words. However, the actual tag to be assigned for an occurrence in text depends on the context of use. We capture this information by using a first-order HMM tagger model. For the transitional probabilities, we begin by using WSJ-based probabilities as a starting point and then adjust to the new domain by using a domain specific text and using EM training. EM also allows for adjusting lexical probabilities derived using WSJ words as exemplars. We call the domain specific text used for training of our HMM tagger as ***Text-EM***. While this could be the same as ***Text-Lex***, we distinguish the two since ***Text-EM*** could be smaller than ***Text-Lex***. From ***Text-Lex***, we only extract a list of words and their frequency of occurrences. In contrast, we use ***Text-EM*** as a text and hence as a sequence of words.

In this work, the set of suffixes that we use is adapted those found in a GRE preparation webpage (DeForest, 2000). A few additional suffixes were obtained from the online English Dictionary AllWords.com (2005). In the future we expect to consider automatic mining of useful suffixes from a domain. Furthermore, prefixes are also useful for our purposes. However apart from a few prefixes used in hyphenated words, we haven't yet incorporated prefix information in a systematic way into our framework.

In this paper, our evaluation domain is molecular biology. Large amounts of text are easily available in the form of Medline abstracts. We use only about 1% of the Medline text database for ***Text-Lex***. Another reason for selecting this evaluation domain is that we have a considerable amount POS-annotated text in this domain, and the most recent techniques of supervised POS tag learning have been used in developing taggers for this domain. This allows us to evaluate our tagger using the annotated text for evaluation as well as to compare our tagger with others developed for this domain. The POS-annotated text we use is the well-known GENIA (Tateisi et al, 2003) corpus that was developed at University of Tokyo.

3 Expectation Maximization Training

Our tagger is a first-order Hidden Markov Model (HMM) tagger that is trained using Expectation Maximization (EM) since we do not assume existence of annotated data in the new domain.[1] Although we use the GENIA corpus, we take only the raw text and strip off the annotated information for obtaining the ***Text-EM***. Our HMM is based on bigram modeling and hence our transitional probabilities correspond to P(t | t') where t and t' are POS tags. The emissions that label the transition edges will be discussed in the next section and include domain words as well as certain types of "coded words".

The initial transitional probabilities are not randomly chosen but rather taken from the WSJ corpus. If we take the transitional probabilities as a representation of syntactic preferences, then EM learning using ***Text-EM*** may be taken as adjustment of the grammatical preferences in the WSJ corpus to those in the new domain. In order to adjust the grammatical preference to the new domain, we start from smoothed WSJ bigram probabilities. If we started from unsmoothed WSJ bigram probabilities, then EM would not allow us to account for transitions that are not observed in the WSJ corpus. For example, in scientific text, transition from RRB (the right round bracket) to VBG may be possible, while it does not occur in the WSJ corpus. Hence, we smooth the WSJ bigram probabilities with WSJ unigram probabilities.

We compute smoothed initial bigram probabilities as

$$P(t \mid t') = \lambda\, P_{WSJ}(t \mid t') + (1-\lambda)\, P_{WSJ}(t),$$

where λ=0.9. We felt employing techniques suggested in (Brants, 2000) gave too high a preference for unigram probabilities.

The initial emit probability is obtained from the domain text ***Text-Lex***. The process is described in the next section. This information is derived purely from suffix and suffix distribution, or from orthographic information and does not account for the actual context of occurrences in the domain text. We take this suffix-based (and orthographic-based) emit probabilities as reasonable initial lexical probabilities. EM training will adjust them as necessary.

We made one minor modification to the standard forward-backward EM algorithm. We dampen the change in transitional and emit probabilities for each iteration. Significant differences in lexical probabilities between the new domain and WSJ can make undue changes in transitional probabilities and this in turn can further lead the lexical probabilities to head in the wrong direction. By adding a damping factor, we can prevent the unsupervised training to spiral out of control. Hence we let the new transitional probability be given by

$$P(t \mid t') = \lambda\, P_{NEW}(t \mid t') + (1-\lambda)\, P_{OLD}(t \mid t')$$

where P_{OLD} represents the transitional probability in the previous iteration and P_{NEW} represents the probability by standard use of forward-backward algorithm. We use a damping factor of 0.5 for both transitional and emit probabilities. For the emit probabilities, this has the effect of moderating POS

[1] We considered a 2nd order model as well, but early work showed negligible advantage predicting to the same training set. Following Wang and Schuurmans (2005) we chose to focus on quality of estimation over model complexity.

preferences derived from the training data and preserving words and POSes from the lexicon for use in the test set.

Even with the damping factor, EM learning followed the pattern of 'Early Maximum' described by Elworthy (1994), where with good initial estimates EM learning only improves accuracy for a few iterations. For our EM training, we fixed iteration 2 as our 'best' EM trained model.

4 Development of the Lexicon and Initial Probabilities

As noted earlier, we use a domain text, ***Text-Lex***, to develop the initial lexical probabilities for the HMM. The essential process is as follows. Let a word w appear a sufficient number of times in ***Text-Lex*** (at least 5 times). We look in ***Text-Lex*** for related words in order to assign a feature vector with this word. Each feature is written as –x+y, where x and y represent suffixes or the empty string (here represented as _).

Features: The feature –x+y represents the word formed by replacing some suffix x in w by some suffix y. Consider the word "creation". "–ion+_" corresponds to the stem word "create" and "–ion+ion" corresponds to the word "creation" itself. The feature "–ion+ed" captures information about the word "created" whereas the feature "-_+s" corresponds to word "creations".

Now consider a word like "history". While this might have non-zero values for "-y+ic" (historic) or "-_+s" (histories), we are likely to set zero value for "–ory+_" (unless "hist" or "histe" is found in ***Text-Lex***). This zero value represents the fact that although "history" has "ory" as a suffix, it has no stem. Such a distinction (whether or not there is a stem) bears much information for suffixes like "ate" and "ory".

We use suffix classes rather than actual suffixes as we believe this provides a more appropriate level of abstraction. Given a word w with a suffix x (for a word with no suffix from our list of suffixes, x is taken to be _. i.e., empty string), we examine whether removal of x from w leads to another word by using a few basic variations that can be found in any rudimentary exposition on English morphology. For example, for the suffix *"ed*", we attempt to replace *"ied"* with *"y"* which relates *"*purified*"* with *"*purify*"* and recognizes the spelling alternation of i/y. Thus for the word *"*purify*"* the feature *"*-+ed" represents the presence of *"*purified*"* since "+ed" represents the suffix class rather than the actual suffix. Similarly, we also consider removal of a suffix and, if necessary, adding an "e" to see if such a word exists. This allows us to relate *"*creation*"* with *"*create*"* or *"*activate*"* with *"*active*"*. Also doubling of a few consonants is attempted to relate *"*occurrence*"* and *"*occur*"*. Finally, when a word could have two suffixes, the word is considered to always have the longer functional suffix. Hence, we consider *"*government*"* to have "ment" suffix rather than "ent" suffix.

Feature Vectors: There are two different types of vectors we use for any word, one called ***Bin*** (for binary count) and other called ***RFreq*** (for relative frequency). In the ***Bin*** vector associated with *"creation",* all these four features will get the value one, assuming that the four corresponding words are found in ***Text-Lex***. On the other hand, assuming *"creatory"* is not found in ***Text-Lex***, the feature "-ion+ory" would get a zero value.

For ***RFreq*** vector, instead of ones and zeros, we first start with the frequency of occurrences of each word and then normalize so that the sum of all feature values is one. Thus, for example, a word with 4 features having non-zero frequencies of 10, 20, 30 and 40 will have the respective values set to 0.1, 0.2, 0.3 and 0.4. A word with four features having non-zero frequency, which are 1, 2, 3 and 4, will also have same 4 relative frequency values.

Our intuition is that the ***Bin*** vector is helpful in determining the set of tags that can be associated with a word and that the ***RFreq*** vector can augment this information regarding the likelihood of these tags. For example, a one for the "-ing+_" feature in a ***Bin*** vector (thus disqualifying a word like "during") may be sufficient to predict VBG, JJ and NN tags. However, this may not suffice to provide the ordering of likelihood among these tags for this word. On the other hand, it seems to be the case that when the "ing" form appears far more often than the "ed" form, then the NN tag is most likely. But if the "ed" form is more frequent, then VBG is most likely. Examples in the WSJ corpus include "smoking", "marketing", "indexing", and "restructuring" for the first kind, and "calling", "counting", "advising", and "noting" for the second kind.

Exemplars in WSJ: Given a word w from ***Text-Lex***, we look for similar words from the WSJ corpus. Even though the set of words used in this corpus may differ substantially from the domain text, our hypothesis is that words with similar suffix distribution will have similar POS tag assignments regardless of the domain. We follow Cucerzan and Yarowsky (2000) in using the kNN method for finding similar words, but we differ in details of the construction of the feature vectors and distance computation. For the word w we create the ***Bin*** and ***RFreq*** vectors based on distribution of words in ***Text-Lex***. Following the same method, we create the ***Bin*** and ***RFreq*** vectors for a word v in the WSJ corpus by using the distributions in the WSJ corpus. Then we compute ***BinDist***(w,v) as the number of features in which the two ***Bin*** vectors differ. A similar ***RFDist*** is defined as a weighted sum of two distances: the first distance is L1-norm distance based on values of features for which both words have non-zero values for and the second distance is based on values of features for which one word has a zero value and other does not. Thus, if the two words' ***RFreq*** vectors are $< w_1,...,w_n >$ and $< v_1,...,v_n >$ respectively then

$$RFsame(w,v) = \sum_{w_i \neq 0 \cap v_i \neq 0} |w_i - v_i|$$

$$RFdiff(w,v) =$$

$$\sum_{w_i = 0 \cap v_i \neq 0} |w_i - v_i| + \sum_{w_i \neq 0 \cap v_i = 0} |w_i - v_i|$$

and,

$$RFDist(w,v) = RFsame(w,v) + \delta RFdiff(w,v)$$

For RFDist(.), we used δ =2. Given a word w, we find the 5 nearest neighbors from the WSJ corpus and use their average lexical probabilities to obtain the lexical probabilities for w. We investigate the use of ***Bin*** vector information and ***RFreq*** vector information for computing the distances (i.e., ***BinDist***(.) and ***RFDist***(.)) as well as a hybrid measure that combines these two distances.

We also considered smoothing the lexical probabilities obtained in the above fashion. Let w be a word for which the above method suggests tags $t_1,\ldots,t_n$ in order of likelihood (t_1 is most probable). Then we consider sqrt-score(t_i)= $\sqrt{n+1-i}$. We then assign probabilities based on this score after normalizing them so that the probabilities for the n tags will sum to 1. Thus, for example, if a word w has three possible tags, no matter what the original lexical probabilities were determined to be, if t_1 is determined to be most probable, then $P(t_1|w)$ will be 0.418 by this method. The second most probable tag will be assigned 0.341.

The intuition behind this square root smoothing method is that this smoothing may be appropriate for low frequency words, where empirical probabilities based purely on a kNN basis may not be entirely appropriate if the new domain is very different. The drawback of course is that if there is sufficient information, we lose useful information by such flattening. And when a tag is significantly more probable for a word then we lose this vital information. For example, the word "high" is mostly annotated as JJ in WSJ corpus but RB and NN are also possible. Square root smoothing will flatten this distribution considerably. Nevertheless, we wish to investigate whether this method of smoothing the distribution is enough in conjunction with EM. EM adjusts the probability from observing the number and context of occurrences in the domain text.[2]

Coded Words: No matter how large ***Text-Lex*** is, there will be words that do not appear a sufficient number of times (we take this number to be 5). We aggregate such words according to their suffixes, if they correspond to one of the predefined suffixes. Then each word with suffix x is considered to be an instance of a "coded" word ***SFX-x***. If a word does not have any of these suffixes then they fall into the coded class ***unknown***. For each such coded word, we assign the tags and probabilities based on similarly aggregated words in the WSJ corpus.

We have two other broad classes of words that we treat differently. Coded words are formed based on orthographic characteristics, which include but are not limited to Greek letters, Roman numerals, digits, upper or lower case single letters, upper case letter sequences, cardinals, certain prefix words, and their combinations. Since they are relatively easy to tag, we do not use the WSJ corpus for them but handle it programmatically. Finally, if a word occurs often in WSJ or is assigned tags such as CD, FW, MD, PRP, DT, WDT, etc. (tags which can't be predicted by means of suffix or suffix-related words), we add this word together with the tags and probability into the domain lexicon that we are building.

[2] We also considered linear and square functions for smoothing while reporting only the sqrt results in section 5.

5 Evaluation and Analysis

As noted earlier, our evaluation is on molecular biology text. For ***Text-Lex***, we used 133,666 titles/abstracts of research papers, a small fraction of the Medline database available from the National Library of Medicine. These abstracts were contained in just five of the 500 compressed data files in the 2006 version of the Medline database. These abstracts cover topics more broadly in Biomedicine and not just molecular biology. On the other hand, we use for ***Text-EM***, text which can be regarded to be in a subfield of molecular biology.

Text-EM is the text from the GENIA corpus (version 3.02) described in (Tateisi et al. 2003). This corresponds to about 2000 abstracts, which are annotated with POS tag information (using the same tags used in the WSJ corpus). We use a 5-fold cross-validation, i.e., 5 partitions are formed and experiments conducted 5 times and results averaged. For each test partition, the remainder partitions are used for "training". In our case, this is unsupervised since we use EM and hence we totally disregard the POS tag information that is associated with the words. We note that both the text for EM training as well as for testing come from the same domain.

We first evaluate the process of building the lexicon. This time we consider the entire GENIA corpus and not any partition. We first considered all words in the GENIA corpus for which we can expect our kNN method to assign a tag. Hence all words that would be treated as coded words are ignored. For each such word, we consider the tags assigned to them in the GENIA corpus and form pairs <w,t>. We are interested in the word type and not token and hence we will not have any multiple occurrences of a pair <w,t>. Our kNN method identifies 96.3% of these pairs; we can think of this as recall. This makes our approach effective, especially given the fact that the kNN method only assigns 1.92 tags on an average to these words in the GENIA corpus. Next considering all words appearing in the GENIA corpus, our lexicon includes a correct tag in 99.0% of the cases on a word-token basis. These results are summarized below.

Characteristic	**Statistic**
kNN Recall (word-type)	96.3%
Average Number Tags/Word	1.92 tags
Lexicon Recall (word-token)	99.0%

We now turn to the evaluation of the accuracy of our HMM. As mentioned earlier, these results are based on 5-fold cross-validation experiments. The best results (95.77%) were obtained for the case where we took the lexical probabilities directly from kNN using only ***RFDist*** and by discarding all tags assigned with probability less than 0.02.[3]

These results compare favorably to other taggers developed for the Biology domain. The MedPost tagger (see Section 6) achieved an accuracy of 94.1% when we applied it to the GENIA abstracts. The PennBioIE tagger (see Section 6) achieved an accuracy of 95.1%. Note that output from the PennBioIE tagger is not fully compatible with GENIA annotation due to some differences in its tokenization. Even if the differences in accuracies can be discounted due to tokenization or even systematic differences in annotation between the training and test corpora, our main point is that our results compare favorably (our tagger competitive) with taggers that were developed for the Biomedicine domain using supervised training.

These results are summarized in the table below.

POS Tagger	**%Accuracy**
Our HMM (5-fold)	95.77%
MedPost	94.1%
PennBioIE	95.1%
GENIA supervised	98.26%

MedPost seems intended to cover all of Biomedicine, since its lexicon is based on the 10,000 most frequently occurring words from Medline and for which the set of possible tags were manually specified. The PennBioIE tagger was developed using 315 Medline abstracts using another subfield of molecular biology.

None of these accuracies however are as high as those of the GENIA tagger (Tsuruoka et al. 2005) which was trained (supervised) using GENIA corpus and uses a machine learning model more sophisticated than the simple first-order HMM tagger we use. This model considers more features including words to the right. The best results (98.26%) were obtained when lexicon from three different sources were aggregated.

[3] Banko and Moore (2004) showed only slight improvement in tag accuracy between .01 and .1 cutoffs with a lexicon built from annotated data. We opted for the .02 cutoff because of our 'noisier' lexicon.

Returning to the results for our taggers, we also tried ***BinDist*** in the kNN method, with and without square root smoothing. These results were typically less than the above-mentioned result. We also compared using a square root smooth on ***RFDist*** obtaining results approximately 1% lower than without the square root smooth.

We next present some examples that illustrate strengths and weaknesses of the current model. An example that shows that EM training makes good adjustment to the domain is the improvement in tagging of verbal categories. We conducted a detailed error analysis on one of the cross-validation partitions and noted that the accuracy on all verbal POS tags improved after EM training. A noteworthy case is the improvement in tagging of VBP originally misclassified as VB. Since most English words that are VB can also be VBP, and since they are annotated more frequently in WSJ as VB, the initial lexicon usually has a higher probability assigned to VB for most words. As EM training progresses, we noted that the frequency of VBP mistagged as VB decreases. Similarly, misclassifications of VBG as NN also drops in the final model (by 40.3% on ***Text-EM***) as compared to the initial model based on WSJ transitional probabilities and initial lexicon derived using WSJ words as exemplars.

Previously, in the context of parsing Biomedical text, Lease and Charniak (2004) mention the occurrences of sequences of multiple NN is more frequent in the GENIA corpus than in the WSJ corpus and that it could lead to parsing errors. We didn't observe this problem here, but rather the contrary situation where many JJs were initially mistagged as NN. About 22% of these misclassifications are corrected after EM training.

While our model adjusts well in these cases to the new domain, sometimes the drift leads to worse performance. An example is in the misclassification of VBN as JJ. The most frequent word for which this misclassification occurs in the word "activated". These misclassifications occur in the context such as "the activated cells". The use of VBN rather than JJ is hard to determine on basis of just surface features and perhaps has to do more with the meaning of the word. In supervised setting, if sufficient such cases were annotated then this would be learned. But in an unsupervised setting this turns out to be a problem case. Despite the fact that ***RFDist*** predicted VBN as most probable tag for "activated", EM training makes this situation worse.

Analysis of words with most frequent errors revealed many cases from orthographic coded words. Many occurrences of single lower case letters (which could have LS, SYM or NN tags) were labeled as LS whereas the GENIA tagging used NN. Our model tagged "+/-" always as SYM whereas because of the context of use, GENIA annotations were CC. (In fact, GENIA does not appear to use the SYM tag.) Similarly, "<" and ">" were often mistagged as SYM by our model whereas based on context they are annotated as JJR.

6 Related Work

The impact of out-of-vocabulary words on NLP applications has been noted before. The degradation in performance of components, which were trained on the WSJ corpus, but used on biomedical text has been noted (Lease and Charniak, 2004, Smith et al, 2005). Smith et al. (2005) use this observation in the design of their POS tagger, MedPost, by building a Markov model with a lexicon containing the 10,000 most frequent words from Medline, and using annotated text from the Biomedical text for supervised training.

There are many unsupervised approaches to POS tagging. We focus now on those that are most closely related to our work and contain ideas that have influenced this work. There have been many uses of EM training to build HMM taggers (Kupiec, 1992; Elworthy, 1994; Banko and Moore, 2004; Wang and Schuurmans, 2005). Banko and Moore (2004) achieved better accuracy by restricting the set of possible tags that are associated with words. By eliminating possibilities that may appear rarely with a word, they reduce the chances of unsupervised training spiraling along an unlikely path. We believe by using our approach we considerably reduce the set of tags to what is appropriate for each word. Further, we too remove any tag associated with low probability by kNN method. Usually these tags are noise introduced by some inappropriate exemplar.

Wang and Schuurman (2005) suggest that EM algorithm be modified such that at any iteration the unigram tag probability be held constant to the true probability for each tag. Again, this might serve to stop a drift in unsupervised methods towards making a tag's probability become larger than it should be. However, the true probability cannot be known ahead of time and certainly not in a new domain. While a WSJ bigram probability need not reflect the corresponding preferences in the new domain, our use of starting from WSJ probabilities and then damping changes to transition probabilities was motivated by a similar concern of not letting a drift towards making some (bigram) tags too frequent during EM iterations.

Using suffixation patterns for purposes of predicting POS tags has been considered before. Although as far as we know, we are the first to apply it for domain adaptation purposes. Schone and Jurafsky (2001) consider clusters of words (obtained by some "perfect" clustering algorithm) and then compute a measure of how "affixy" a cluster is. For example, a cluster containing words "climb" and "jump" may be related by suffixing operation +s to a cluster that contains words "climbs" and "jumps". The percentage of words in a cluster that are so related provides a measure of how "affixy" a cluster. This together with five other attributes of clusters (such as whether words in a cluster precede those of another cluster, optionality) and language universals induce POS tags for these clusters from corpora. This method does not use POS tagged corpora (although in the reported experiment the initial "perfect" clusters were obtained from the Brown corpus using the POS tag information). In contrast, we use the POS tagged WSJ corpus to assist in the induction of tag information for our lexicon. In this respect, our method is closer to the approach of Cucerzan and Yarowsky (2000). Our use of the kNN method to identify tags and their probabilities for words was inspired by this work. However, their use of kNN method was in the context of supervised learning. The method was applied for handling words unseen in the training data. The estimated probabilities were used during the tagging process. Instead of just applying the method for unknown words, i.e., words not present in the training data, our approach is to create the entire lexicon in the new domain. As Lease and Charniak (2004), among others, have noted, the distribution of NN tag sequences as well as tag distributions in the Biomedical domain could differ from WSJ text. Since our aim is to adjust to the new domain, we employed unsupervised learning in the form of EM training, unlike the supervised tagging model development approach of Cucerzan and Yarowsky. Another significant difference is that their method determines nearest neighbors not only on the basis of suffix-related words but also on the basis of nearby words context. Since our motivation, on the other hand, is to move to a new domain, we didn't consider detection of similarity on the basis of word contexts. In contrast, we have shown that the approach of identifying words on the basis of suffixation patterns and using them as exemplars can be applied effectively even when the domain of application is substantially different from the text (the WSJ corpus) providing the exemplars.

7 Conclusions

As NLP technology continues to be applied in new domains, it becomes more important to consider the issue of portability to new domains. To cope with domain-specific vocabulary and also different use of vocabulary in a new domain, we exploited suffix information of words. While use of suffix information per se has been employed in many existing POS taggers, its use is often limited to an online manner, where each word is examined independently from the existence of its morphologically related words. As shown in (Cucerzan and Yarowsky, 2000), such information can provide considerable information to build a lexicon that associates possible tags with words. However, we use this information only to provide the initial values. We apply EM algorithm to adjust these initial probabilities to the new domain.

The results in Section 5 show that we achieve good performance in the evaluation domain, which is comparable with two recently developed taggers for this domain. We also show in section 5 examples of how EM unlearns some WSJ bias and adjusts to the new domain. While we introduce a damping factor to slow down changes in iterations of EM training, we believe there is scope for further improvement to minimize drift. Furthermore, there is scope to improve our kNN method as discussed at the end of Section 5. In the future, we also expect to consider methods that may automatically mine suffixes in a new domain and use these domain-specific suffixes. We used the kNN method to associate words in the new domain with possible POS tags.

Despite the often-stated notion that English is not morphologically rich, we find that suffix-based methods can still help make significant inroads.

Our method offers the chance to develop good taggers for specialized domains. For example, the GENIA corpus and PennBioIE corpus are specializations within molecular biology, but taggers developed on one corpus degrades in performance on the other. Using our method, we could use different ***Text-EM*** for these specializations even if we retain Medline as ***Text-Lex***. In the same way, we could develop a tagger for the medical domain, which has a distinct vocabulary from biology.

References

Banko, M. and Moore, R.C. 2004. Part of Speech Tagging in Context. In Proceedings, 20th International Conference on Computational Linguistics (Coling 2004), Geneva, Switzerland, pp.556-561.

Baum, L. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. Inequalities 3:1-8.

Brants, T. 2000. TNT - a statistical part-of-speech tagger. In Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000.

Brill, E. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. Computational Linguistics, 21(4):543-565.

Cucerzan, S. and Yarowsky, D. 2000. Language independent minimally supervised induction of lexical probabilities. Proceedings of ACL-2000, Hong Kong, pages 270-277.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. 1992. A practical part of sppech tagger. Proceedings of 3rd Conference on Applied Natural Language Processing, 53-58.

DeForest, J. 2000. Graduate Record Exam Suffixed web page. Michigan State University. http://www.msu.edu/~defores1/gre/sufx/gre_suffx.htm

Elworthy, D. 1994. Does Baum-Welch re-estimation help taggers. In Proceedings of the Fourth Conference on Applied Natural Language Processing, ACL.

Kulick, S., Bies, A., Liberman, M., Mandel, M., McDonald, R., Palmer, M., Schein, A. and Ungar, L. *Integrated Annotation for Biomedical Information Extraction.* HLT/NAACL, 2004.

Kupiec, J. 1992. Robust Part-of-speech Tagging Using a Hidden Markov Model. Computer Speech and Language, 6.

Lease, M. and Charniak, E. 2005. Parsing Biomedical Literature. IJCNLP-2005: 58-69.

Marcus, M., Santorini, B., Marcinkiewicz, M.A. 1993. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, 19:313-330.

PennBioIE. 2005. Mining The Bibliome Project. http://bioie.ldc.upenn.edu/.

Schone, P. and Jurafsky, D. 2001, Language Independent Induction of Part of Speech Class Labels Using Language Universals. IJCAI Workshop on Text Learning: Beyond Supervision.

Smith, L., Rindflesch, T., Wilbur, W.J. 2004. MedPost: a part-of-speech tagger for biological text. Bioinformatics 20 (14):2320-2321.

Smith, L., Rindflesch, T., Wilbur, W.J. 2005. The importance of the lexicon in tagging biomedical text. Natural Language Engineering 12(2) 1-17.

Tateisi, Y., Ohta, T., dong Kim, J., Hong, H., Jian, S., Tsujii, J. 2003. The GENIA corpus: Medline abstracts annotated with linguistic information. In: Third meeting of SIG on Text Mining, Intelligent Systems for Molecular Biology (ISMB).

Tsuruoka, Y., Tateishi, Y., Kim, J. D., Ohta, T., McNaught, J., Ananiadou, S., and Tsujii, J.. 2005. Developing a Robust Part-of-Speech Tagger for Biomedical Text, Advances in Informatics - 10th Panhellenic Conference on Informatics, LNCS 3746: 382-392.

Wang, Q. and Schuurmans, D. 2005. Improved estimation for unsupervised part-of-speech tagging. In IEEE NLP-KE

www.AllWords.com. 2005. English Dictionary and Language Guide. AllSites LLC. www.AllSitesllc.com

Multilingual Dependency Parsing and Domain Adaptation using DeSR

Giuseppe Attardi
Felice Dell'Orletta
Maria Simi
Dipartimento di Informatica
largo B. Pontecorvo 3
I-56127 Pisa, Italy
attardi@di.unipi.it
felice.dellorletta@ilc.cnr.it
simi@di.unipi.it

Atanas Chanev
Università di Trento
via Matteo del Ben 5
I-38068 Rovereto, Italy
Fondazione Bruno Kessler-irst
via Sommarive 18
I-38050 Povo, Italy
chanev@form.unitn.it

Massimiliano Ciaramita
Yahoo! Research Barcelona
Ocata 1
S-08003 Barcelona, Spain
massi@yahoo-inc.com

Abstract

We describe our experiments using the DeSR parser in the multilingual and domain adaptation tracks of the CoNLL 2007 shared task. DeSR implements an incremental deterministic Shift/Reduce parsing algorithm, using specific rules to handle non-projective dependencies. For the multilingual track we adopted a second order averaged perceptron and performed feature selection to tune a feature model for each language. For the domain adaptation track we applied a tree revision method which learns how to correct the mistakes made by the base parser on the adaptation domain.

1 Introduction

Classifier-based dependency parsers (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004) learn from an annotated corpus how to select an appropriate sequence of Shift/Reduce actions to construct the dependency tree for a sentence. Learning is based on techniques such as SVM (Vapnik 1998) or Memory Based Learning (Daelemans 2003), which provide high accuracy but are often computationally expensive. For the multilingual track in the CoNLL 2007 Shared Task, we employed a Shift/Reduce parser which uses a perceptron algorithm with second-order feature maps, in order to verify whether a simpler and faster algorithm can still achieve comparable accuracy.

For the domain adaptation track we wished to explore the use of tree revisions in order to incorporate language knowledge from a new domain.

2 Multilingual Track

The overall parsing algorithm is a deterministic classifier-based statistical parser, which extends the approach by Yamada and Matsumoto (2003), by using different reduction rules that ensure deterministic incremental processing of the input sentence and by adding specific rules for handling non-projective dependencies. The parser also performs dependency labeling within a single processing step.

The parser is modular and can use several learning algorithms. The submitted runs used a second order Average Perceptron, derived from the multiclass perceptron of Crammer and Singer (2003).

No additional resources were used. No preprocessing or post-processing was used, except stemming for English, by means of the Snowball stemmer (Porter 2001).

3 Deterministic Classifier-based Parsing

DeSR (Attardi, 2006) is an incremental deterministic classifier-based parser. The parser constructs dependency trees employing a deterministic bottom-up algorithm which performs Shift/Reduce actions while analyzing input sentences in left-to-right order.

Using a notation similar to (Nivre and Scholz, 2003), the state of the parser is represented by a

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1112–1118,
Prague, June 2007. ©2007 Association for Computational Linguistics

quadruple $\langle S, I, T, A\rangle$, where S is the stack of past tokens, I is the list of (remaining) input tokens, T is a stack of temporary tokens and A is the arc relation for the dependency graph.

Given an input string W, the parser is initialized to $\langle (), W, (), ()\rangle$, and terminates when it reaches a configuration $\langle S, (), (), A\rangle$.

The three basic parsing rule schemas are as follows:

$$\textit{Shift} \quad \frac{\langle S, n|I, T, A\rangle}{\langle n|S, I, T, A\rangle}$$

$$\textit{Right}_d \quad \frac{\langle s|S, n|I, T, A\rangle}{\langle S, n|I, T, A\cup\{(s, d, n)\}\rangle}$$

$$\textit{Left}_d \quad \frac{\langle s|S, n|I, T, A\rangle}{\langle S, s|I, T, A\cup\{(n, d, s)\}\rangle}$$

The schemas for the *Left* and *Right* rules are instantiated for each dependency type $d \in D$, for a total of $2|D| + 1$ rules. These rules perform both attachment and labeling.

At each step the parser uses classifiers trained on a treebank corpus in order to predict which action to perform and which dependency label to assign given the current configuration.

4 Non-Projective Relations

For handling non-projective relations, Nivre and Nilsson (2005) suggested applying a preprocessing step to a dependency parser, which consists in lifting non-projective arcs to their head repeatedly, until the tree becomes pseudo-projective. A post-processing step is then required to restore the arcs to the proper heads.

In DeSR non-projective dependencies are handled in a single step by means of the following additional parsing rules, slightly different from those in (Attardi, 2006):

$$\textit{Right2}_d \quad \frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle\ S, s_1|n|I, T, A\cup\{(s_2, d, n)\}\rangle}$$

$$\textit{Left2}_d \quad \frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle s_2|S, s_1|I, T, A\cup\{(n, d, s_2)\}\rangle}$$

$$\textit{Right3}_d \quad \frac{\langle s_1|s_2|s_3|S, n|I, T, A\rangle}{\langle\ S, s_1|s_2|n|I, T, A\cup\{(s_3, d, n)\}\rangle}$$

$$\textit{Left3}_d \quad \frac{\langle s_1|s_2|s_3|S, n|I, T, A\rangle}{\langle s_2|s_3|S, s_1|I, T, A\cup\{(n, d, s_3)\}\rangle}$$

$$\textit{Extract} \quad \frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle n|s_1|S, I, s_2|T, A\rangle}$$

$$\textit{Insert} \quad \frac{\langle S, I, s_1|T, A\rangle}{\langle s_1|S, I, T, A\rangle}$$

Left2, *Right2* are similar to *Left* and *Right*, except that they create links crossing one intermediate node, while *Left3* and *Right3* cross two intermediate nodes. Notice that the *RightX* actions put back on the input the intervening tokens, allowing the parser to complete the linking of tokens whose processing had been delayed. *Extract*/*Insert* generalize the previous rules by moving one token to the stack T and reinserting the top of T into S.

5 Perceptron Learning and 2nd-Order Feature Maps

The software architecture of the DeSR parser is modular. Several learning algorithms are available, including SVM, Maximum Entropy, Memory-Based Learning, Logistic Regression and a few variants of the perceptron algorithm.

We obtained the best accuracy with a multiclass averaged perceptron classifier based on the ultraconservative formulation of Crammer and Singer (2003) with uniform negative updates. The classifier function is:

$$F(x) = \arg\max_k \{\alpha_k \cdot x\}$$

where each parsing action k is associated with a weight vector α_k. To regularize the model the final weight vectors are computed as the average of all weight vectors posited during training. The number of learning iterations over the training data, which is the only adjustable parameter of the algorithm, was determined by cross-validation.

In order to overcome the limitations of a linear perceptron, we introduce a feature map Φ: $\mathbb{R}^d \to \mathbb{R}^{d(d+1)/2}$ that maps a feature vector $\boldsymbol{x}$ into a higher dimensional feature space consisting of all unordered feature pairs:

$$\Phi(\boldsymbol{x}) = \langle x_i x_j \mid i = 1, \ldots, d, j = i, \ldots, d\rangle$$

In other words we expand the original representation in the input space with a feature map that generates all second-order feature combinations from each observation. We call this the 2nd-order model, where the inner products are computed as $\alpha_k \cdot \Phi(\boldsymbol{x})$, with α_k a vector of dimension $d(d+1)/2$. Applying a linear perceptron to this feature space corresponds to simulating a polynomial kernel of degree two.

A polynomial kernel of degree two for SVM was also used by Yamada and Matsumoto (2003). However, training SVMs on large data sets like those arising from a big training corpus was too computationally expensive, forcing them to resort to partitioning the training data (by POS) and to learn several models.

Our implementation of the perceptron algorithm uses sparse data structures (hash maps) so that it can handle efficiently even large feature spaces in a single model. For example the feature space for the 2nd-order model for English contains over 21 million. Parsing unseen data can be performed at tens of sentences per second. More details on such aspects of the DeSR parser can be found in (Ciaramita and Attardi 2007).

6 Tuning

The base parser was tuned on several parameters to optimize its accuracy as follows.

6.1 Feature Selection

Given the different characteristics of languages and corpus annotations, it is worth while to select a different set of features for each language. For example, certain corpora do not contain lemmas or morphological information so lexical information will be useful. Vice versa, when lemmas are present, lexical information might be avoided, reducing the size of the feature set.

We performed a series of feature selection experiments on each language, starting from a fairly comprehensive set of 43 features and trying all variants obtained by dropping a single feature. The best of these alternatives feature models was chosen and the process iterated until no further gains were achieved. The score for the alternatives was computed on a development set of approximately 5000 tokens, extracted from a split of the original training corpus.

Despite the process is not guaranteed to produce a global optimum, we noticed LAS improvements of up to 4 percentage points on some languages.

The set of features to be used by DeSR is controlled by a number of parameters supplied through a parameter file. Each parameter describes a feature and from which tokens to extract it. Tokens are referred through positive numbers for input tokens and negative numbers for tokens on the stack. For example

```
PosFeatures       -2 -1 0 1 2 3
```

means to use the POS tag of the first two tokens on the stack and of the first four tokens on the input. The parameter *PosPrev* refers to the POS of the preceding token in the original sentence, *PosLeftChild* refers to the POS of the left children of a token, *PastActions* tells how many previous actions to include as features.

The selection process was started from the following base feature model:

```
LexFeatures      -1 0 1
LemmaFeatures    -2 -1 0 1 2 3
LemmaPrev        -1 0
LemmaSucc        -1 0
LemmaLeftChild   -1 0
LemmaRightChild  -1
MorphoFeatures   -1 0 1 2
PosFeatures      -2 -1 0 1 2 3
PosNext          -1 0
PosPrev          -1 0
PosLeftChild     -1 0
PosRightChild    -1 0
CPosFeatures     -1 0 1
DepFeatures      -1 0
DepLeftChild     -1 0
DepRightChild    -1
PastActions      1
```

The selection process produced different variants for each language, sometimes suggesting dropping certain intermediate features, like the lemma of the third next input token in the case of Catalan:

```
LemmaFeatures    -2 -1 0 1 3
LemmaPrev        0
LemmaSucc        -1
LemmaLeftChild   0
LemmaRightChild  -1
PosFeatures      -2 -1 0 1 2 3
PosPrev          0
PosSucc          -1
PosLeftChild     -1 0
PosRightChild    -1 0
CPosFeatures     -1 0 1
MorphoFeatures   0 1
DepLeftChild     -1 0
DepRightChild    -1
```

For Italian, instead, we ran a series of tests in parallel using a set of manually prepared feature models. The best of these models achieved a LAS of 80.95%. The final run used this model with the addition of the morphological agreement feature discussed below.

English was the only language for which no feature selection was done and for which lexical features

Task	LAS			UAS		
	1st	DeSR	Avg	1st	DeSR	Avg
Arabic	76.52	72.66	68.34	86.09	82.53	78.84
Basque	76.92	69.48	68.06	82.80	76.86	75.15
Catalan	88.70	86.86	79.85	93.40	91.41	87.98
Chinese	84.69	81.50	76.59	88.94	86.73	81.98
Czech	80.19	77.37	70.12	86.28	83.40	77.56
English	89.61	85.85	80.95	90.63	86.99	82.67
Greek	76.31	73.92	70.22	84.08	80.75	77.78
Hungarian	80.27	76.81	71.49	83.55	81.81	76.34
Italian	84.40	81.34	78.06	87.91	85.54	82.45
Turkish	79.81	76.87	73.19	86.22	83.56	80.33

Table 1. Multilingual track official scores**.**

were used. English is also the language where the official score is significantly lower than what we had been getting on our development set (90.01% UAS).

6.2 Prepositional Attachment

Certain languages, such as Catalan, use detailed dependency labeling, that for instance distinguish between adverbials of location and time. We exploited this information by introducing a feature that captures the entity type of a child of the top word on the stack or in the input. During training a list of nouns occurring in the corpus as dependent on prepositions with label CCL (meaning ‘complement of location’ for Catalan) was created and similarly for CCT (complement of time). The entity type TIME is extracted as a feature depending on whether the noun occurs in the time list more than α times than in the location list, and similarly for the feature LOCATION. α was set to 1.5 in our experiments.

6.3 Morphological Agreement

Certain languages require gender and number agreement between head and dependent. The feature `MorphoAgreement` is computed for such languages and provided noticeable accuracy improvements.

For example, for Italian, the improvement was from:

LAS: 80.95%, UAS: 85.03%

to:

LAS: 81.34%, UAS: 85.54%

For Catalan, adding this feature we obtained an unofficial score of:

LAS: 87.64%, UAS: 92.20%

with respect to the official run:

LAS: 86.86%, UAS: 91.41%

7 Accuracy

Table 1 reports the accuracy scores in the multilingual track. They are all considerably above the average and within 2% from the best for Catalan, 3% for Chinese, Greek, Italian and Turkish.

8 Performance

The experiments were performed on a 2.4 Ghz AMD Opteron machine with 32 GB RAM. Training the parser using the 2^{nd}-order perceptron on the English corpus required less than 3 GB of memory and about one hour for each iteration over the whole dataset. Parsing the English test set required 39.97 sec. For comparison, we tested the MST parser version 0.4.3 (Mstparser, 2007), configured for second-order, on the same data: training took 73.9 minutes to perform 10 iterations and parsing took 97.5 sec. MST parser achieved:

LAS: 89.01%, UAS: 90.17%

9 Error Analysis on Catalan

The parser achieved its best score on Catalan, so we performed an analysis on its output for this language.

Among the 42 dependency relations that the parser had to assign to a sentence, the largest number of errors occurred assigning *CC* (124), *SP* (33), *CD* (27), *SUJ* (26), *CONJUNCT* (22), *SN* (23).

The submitted run for Catalan did not use the entity feature discussed earlier and indeed 67 errors were due to assigning CCT or CCL instead of CC (generic complement of circumstance). However over half of these appear as underspecified annotation errors in the corpus rather than parser errors.

By adding the `ChildEntityType` feature, which distinguishes better between CCT and CCL, the UAS improved, while the LAS dropped slightly, due to the effect of underspecified annotations in the corpus:

LAS: 87.22%, UAS: 91.71%

A peculiar aspect of the original Catalan corpus was the use of a large number (195) of dependency labels. These labels were reduced to 42 in the version used for CoNNL 2007, in order to make it comparable to other corpora. However, performing some preliminary experiments using the original Catalan collection with all 195 dependency labels, the DeSR parser achieved a significantly better score:

LAS: 88.80%, UAS: 91.43%

while with the modified one, the score dropped to:

LAS: 84.55%, UAS: 89.38%

This suggests that accuracy might improve for other languages as well if the training corpus was labeled with more precise dependencies.

10 Adaptation Track

The adaptation track originally covered two domains, the CHILDES and the Chemistry domain.

The CHILDES (Brown, 1973; MacWhinney, 2000) consists of transcriptions of dialogues with children, typically short sentences of the kind:

```
Would you like more grape juice ?
That 's a nice box of books .
```

Phrases are short, half of them are questions. The only difficulty that appeared from looking at the unlabeled collection supplied for training in the domain was the presence of truncated terms like *goin* (for *going*), *d* (for *did*), etc. However none of these unusually spelled words appeared in the test set, so a normal English parser performed reasonably well on this task. Because of certain inconsistencies in the annotation guidelines, the organizers decided to make this task optional and hence we submitted just the parse produced by the parser trained for English.

For the second adaptation task we were given a large collection of unlabeled data in the chemistry domain (Kulick et al, 2004) as well as a test set of 5000 tokens (200 sentences) to parse (*english_pchemtbtb_test.conll*).

There were three sets of unlabeled documents: we chose the smallest (*unlab1*) consisting of over 300,000 tokens (11663 sentences). *unlab1* was tokenized, POS and lemmas were added using our version of TreeTagger (Schmid, 1994), and lemmas replaced with stems, which had turned out to be more effective than lemmas. We call this set *pchemtb_unlab1.conll*.

We trained the DeSR parser on English using *english_ptb_train.conll*, the WSJ PTB collection provided for CoNLL 2007. This consists of WSJ sections 02-11, half of the usual set 02-23, for a total of 460,000 tokens with dependencies generated with the converter by Johansson and Nugues (2007).

We added stems and produced a parser called *DeSRwsj*. By parsing *english_pchem_test.conll* with *DeSRwsj* we obtained *pchemtb_test_base.desr*, our baseline for the task.

By visual inspection using DgAnnotator (DgAnnotator, 2006), the parses looked generally correct. Most of the errors seemed due to improper handling of conjunctions and disjunctions. The collection in fact contains several phrases like:

```
Specific antibodies raised against
P450IIB1 , P450 IA1 or IA2 ,
P450IIE1 , and P450IIIA2 inhibited
the activation in liver microsomes
from rats pretreated with PB , BNF ,
INH and DEX respectively
```

The parser did not seem to have much of a problem with terminology, possibly because the supplied gold POS were adequate.

For the adaptation we proceeded as follows. We parsed *pchemtb_unlab1.conll* using *DeSRwsj* obtaining *pchemtb_unlab1.desr*.

We then extracted a set of 12,500 sentences from *ptb_train.conll* and 7,500 sentences from *pchemtb_unlab1.desr*, creating a corpus of 20,000 sentences called *combined.conll*. In both cases the selection criteria was to choose sentences shorter than 30 tokens.

We then trained a low accuracy parser (called *DesrCombined*) on *combined.conll*, by using a 1st-order averaged perceptron. *DesrCombined* was used to parse *english_ptb_train.conll*, the original training corpus for English. By comparing this parse with the original, one can detect where such parser makes mistakes. The rationale for using an inaccurate parser is to obtain parses with many errors so that they form a suitably large training set for the next step: parser revision.

We then used a parsing revision technique (Attardi and Ciaramita, 2007) to learn how to correct these errors, producing a parse reviser called *DesrReviser*. The revision technique consists of comparing the parse trees produced by the parser with the gold standard parse trees, from the annotated corpus. Where a difference is noted, a revision rule is determined to correct the mistake. Such rules consist in movements of a single link to a different head. Learning how to revise a parse tree consists in training a classifier on a set of training examples consisting of pairs $\langle(w_i, d, w_j), t_i\rangle$, i.e. the link to be modified and the transformation rule to apply. Attardi and Ciaramita (2007) showed that 80% of the corrections can be typically dealt with just 20 tree revision rules. For the adaptation track we limited the training to errors recurring at least 20 times and to 30 rules.

DesrReviser was then applied to *pchemtb_test_base.desr* producing *pchemtb_test_rev.desr*, our final submission.

Many conjunction errors were corrected, in particular by moving the head of the sentence from a coordinate verb to the conjunction *'and'* linking two coordinate phrases.

The revision step produced an improvement of 0.42% LAS over the score achieved by using just the base *DeSRwsj* parser.

Table 2 reports the official accuracy scores on the closed adaptation track. DeSR achieved a close second best UAS on the *ptchemtb* test set and third best on *CHILDES*. The results are quite encouraging, particularly considering that the revision step does not yet correct the dependency labels and that our base English parser had a lower rank in the multilingual track.

Task	LAS			UAS		
	1st	DeSR	Avg	1st	DeSR	Avg
CHILDES				61.37	58.67	57.89
Pchemtb	81.06	80.40	73.03	83.42	83.08	76.42

Table 2. Closed adaptation track scores.

Notice that the adaptation process could be iterated. Since the combination *DeSRwsj+DesrReviser* is a more accurate parser than *DeSRwsj*, we could use it again to parse *pchemtb_unlab1.conll* and so on.

11 Conclusions

For performing multilingual parsing in the CoNLL 2007 shared task we employed DeSR, a classifier-based Shift/Reduce parser. We used a second order averaged perceptron as classifier and achieved accuracy scores quite above the average in all languages. For proper comparison with other approaches, one should take into account that the parser is incremental and deterministic; hence it is typically faster than other non linear algorithms.

For the adaptation track we used a novel approach, based on the technique of tree revision, applied to a parser trained on a corpus combining sentences from both the training and the adaptation domain. The technique achieved quite promising results and it also offers the interesting possibility of being iterated, allowing the parser to incorporate language knowledge from additional domains.

Since the technique is applicable to any parser, we plan to test it also with more accurate English parsers.

Acknowledgments. The following treebanks were used for training the parser: (Aduriz et al., 2003; Böhmovà et al., 2003; Chen et al., 2003; Hajiç et al., 2004; Marcus et al., 1993; Martí et al., 2002; Montemagni et al. 2003; Oflazer et al., 2003; Prokopidis et al., 2005; Csendes et al., 2005). Ryan McDonald and Jason Baldridge made available mstparser and helped us using it. We gratefully acknowledge Hugo Zaragoza and Ricardo Baeza-Yates for supporting the first author during a sabbatical at Yahoo! Research Barcelona.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia and M. Oronoz. 2003. Construction of a Basque Dependency Treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, 201–204.

G. Attardi. 2006. Experiments with a Multilanguage non-projective dependency parser. In *Proc. of the Tenth CoNLL, 2006.*

G. Attardi, M. Ciaramita. 2007. Tree Revision Learning for Dependency Parsing. In *Proc. of NAACL/HLTC 2007*.

A. Böhmová, J. Hajic, E. Hajicová and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, 103–127.

R. Brown. 1973. *A First Language: The Early Stages.* Harvard University Press.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang and Z. Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé (2003), chapter 13, 231–248.

M. Ciaramita, G. Attardi. 2007. Dependency Parsing with Second-Order Feature Maps and Annotated Semantic Information. *Proc. of the 12th International Workshop on Parsing Technologies* (*IWPT*), *2007*.

K. Crammer, Y. Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *Journ. of Machine Learning Research*.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

DgAnnotator. 2006. http://medialab.di.unipi.it/Project/Parser/DgAnnotator/.

J. Hajic, O. Smrz, P. Zemánek, J. Snaidauf and E. Beska. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA).*

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL).*

B. MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Lawrence Erlbaum.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez and M. Bertran. 2007. CESS-ECE: A Multilingual and Multilevel Annotated Corpus. Available for download from: http://www.lsi.upc.edu/~mbertran/cess-ece/.

R. McDonald, et al. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proc. of HLT-EMNLP*.

B. MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, 189–210.

Mstparser 0.4.3. 2007. http://sourceforge.net/projects/mstparser/

J. Nivre, et al. 2004. Memory-based Dependency Parsing. In *Proc.s of the Eighth CoNLL*, ed. H. T. Ng and E. Riloff, Boston, Massachusetts, 49–56.

J. Nivre and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proc. of the 43rd Annual Meeting of the ACL*, 99–106.

J. Nivre and M. Scholz. 2004. Deterministic Dependency Parsing of English Text. In *Proc. of COLING 2004*, Geneva, Switzerland, 64–70.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, 261–277.

M.F. Porter. 2001. Snowball Stemmer. http://www.snowball.tartarus.org/

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek depen- dency treebank. In Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT), pages 149–160.

H. Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proc. of International Conference on New Methods in Language Processing*.

V. N. Vapnik. 1998. The Statistical Learning Theory. Springer.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proc. of the 8th International Workshop on Parsing Technologies* (*IWPT*), 195–206.

Hybrid Ways to Improve Domain Independence in an ML Dependency Parser

Eckhard Bick
Institute of Language and Communication
University of Southern Denmark
5230 Odense M, Denmark
eckhard.bick@mail.dk

Abstract

This paper reports a hybridization experiment, where a baseline ML dependency parser, LingPars, was allowed access to Constraint Grammar analyses provided by a rule-based parser (EngGram) for the same data. Descriptive compatibility issues and their influence on performance are discussed. The hybrid system performed considerably better than its ML baseline, and proved more robust than the latter in the domain adaptation task, where it was the best-scoring system in the open class for the chemical test data, and the best overall system for the CHILDES test data.

1 Introduction

LingPars, a language-independent treebank-learner developed in the context of the CoNLL-X 2006 shared task (http://nextens.uvt.nl/~conll/), was inspired by the Constraint Grammar (CG) parsing approach (Karlsson et al. 1995) in the sense that it prioritized the identification of syntactic function over syntactic form, basing the dependency potential of a word on "edge" labels like subject, object etc. rather than the other way around. The system also used other features typical of CG systems, such as BARRIER conditions, tag chains of variable length, implicit clause boundaries and tag sets (Bick 2006). For the 2007 task only one such feature was newly introduced - a *directedness* marker for a few major functions, splitting subject, adverbial and adnominal labels into pairs of left- and right-attaching labels (e.g. SBJ-L, SBJ-R, NMOD-L, NMOD-R). Even this small addition, however, increased the memory space requirements of the model to such a degree that only runs with 50-75% of the training data were possible on the available hardware.

The main purpose of the LingPars architecture changes for CoNLL2007 (Nivre et al. 2007), however, was to test two core hypotheses:

- Can an independent, rule-based parser be made to conform to different, data-imposed descriptive conventions without too great a loss in accuracy?
- Does a rules-based dependency parser have a better chance than a machine-learned one to identify long-distance relations and global sentence structure, thus providing valuable arbiter information to the latter?

Obviously, both points rule out a test involving many languages with the *same* parser (CoNLL task 1). The domain adaptation task (task 2), however, satisfied the single-language condition and also adressed the descriptive adaptation problem (second hypothesis), involving three English treebanks - Wall Street Journal data from the Penn treebank (PTB, Marcus et al. 1993) for training, and the Pchem (Kulick et al. 2004) and CHILDES (Brown 1973 and MacWhinney 2000) treebanks with biomedical and spoken language data, respectively.

2 Developing and adapting EngGram

A parser with hand-written rules pays a high "labour price" to arrive at deep, linguistically pre-

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1119–1123,
Prague, June 2007. ©2007 Association for Computational Linguistics

dictable and versatile analyses. For CG systems as employed by the author, the cost, from lexicon to dependency, is usually several man years, and results are not language-independent. One way of increasing development efficiency is to combine modules for different levels of analysis while reusing or adapting the less-language independent ones. Thus, the development of a new English dependency parser, EngGram, under way for some time, was accelerated for the present project by seeding the syntactic disambiguation grammar with *Danish* rules from the well-established DanGram parser (http://beta.visl.sdu.dk/constraint_grammar.html). By maintaining an identical set of syntactic function tags, it was even possible to use the Danish dependency module (Bick 2005) with only minor adaptations (mainly concerning noun chains and proper nouns).

In order to integrate the output of a CG parser into an ML parser for the shared task data, several levels of compatibility issues have to be addressed. On the input side, (1) PTB tokenization and (2) word classes (PoS) have to be fed into the CG parser bypassing its own modules of morphological analysis and disambiguation. On the output side, (3) CG function categories and (4) attachment conventions have to be adapted to match PTB ones.

For example, the manual rules were tuned to a tokenization system that handles expressions such as "a=few", "at=least" and "such=as" as units. Though amounting to only 1% of running text, they constitute syntactically crucial words, and misanalysis leads to numerous secondary errors. Even worse is the case of the genitive-s (also with a frequency of 1%), tokenised in the PTB convention, but regarded a morpheme in EngGram. Since EngGram does not have a word class for the isolated 's', and since ordinary rules disfavour postnominal singel-word attachment, the 's' had to be fused in PTB-to-CG input, creating fewer tokens and thus problems in re-aligning the analysed output. Also relevant for a full structure parser is the parse window. Here, in order to match PTB window size, EngGram had to be forced not to regard ; ( ) and : as delimiters, with an arguable loss in annotation accuracy due to rules with global NOT contexts designed for smaller windows.

Finally, PTB convention fuses certain word classes, like subordinating conjunctions and prepositions (IN), and the infititive marker and the preposition "to" (TO). Though these cases can be treated by letting CG disambiguation override the CoNLL input's pos tag, input pos can then no longer be said to be "known", with some deterioration in recall as a consequence. Open class categories matched well even at a word-by-word level, closed class tokens were found to sometimes differ for individual words, an error source left largely unchecked.

Treebank error rate is another factor to be considered - in cases where the PoS accuracy of the human-revised treebanks is lower than that of a CG system, the latter should be allowed to *always* assign its own tags, rather than follow the supposedly fixed input pos. In the domain adaptation task, the CHILDES data were a case in point. A separate CG run indicated 6.6% differences in PoS, and manual inspection of part of the cases suggested that while some cases were irrelevant variations (e.g. adjective vs. participle), most were real error on the part of the treebank, and the parser was therefore set to ignore test data annotation and to treat it as pure text.

Errors appeared to be rarer in the training data, but inconsistencies between pos and function label (e.g. IN-preposition and SBJ-subject for "that") prove that errors aren't unknown here either - which is why a hybrid system with independent analysis has the potential benefit of compensating for "mis-learned" patterns in the ML system.

Output conversion from CG to PTB/CoNLL format had to address, besides realignment of tokens (e.g. genitive-s), the disparity in edge (function) labels. However, since the PTB set was more coarse grained, it was possible to simply lump several EngGram labels into one PTB label, for instance:

SC, OC, SUB, INFM --> VMOD
ADVL, SA, OA, PIV, PRED -> ADV

Some idiosyncrasies had to be observed here, for instance the treatment of SC (subject complement) as VMOD for words, but ADV for clauses, or the descriptive decision to tag direct objects in ACI constructions with OA-clausal complements as subjects. Some cases of label variation, however, could not be solved in a systematic way. Thus, adverbs within verb chains, always ADVL in EngGram, could not systematically be mapped, since PTB uses both VMOD and ADV in this position. A certain percentage of mismatches in spite of a correct analysis must therefore be taken into account as part of the "price" for letting the CG system advise the machine learner.

Dependencies were generally used in the same way in both systems, but multi word expressions were problematic, since PTB - without marking them as MWE - appears to attach all elements to a common head even where internal structure (e.g. a PP) is present. No reliable way was found to predict this behaviour from CG dependency output. Finally, PTB often uses the adverbial modifier tag (AMOD) for what would logically be the *head* of an expression:

*about (**head**) 1,200 (AMOD)*
*so (**head**) totally (AMOD)*
*herbicide (**head**) resistant (AMOD)*

EngGram in these examples regards the first element as AMOD modifier, and the second as head. Since the inversion was so common, it was accepted as either intentional or systematically erroneous, and the CG output inverted accordingly. It is an open question, for future research, whether the CG and ML systems could have been harmonized better, had the training data been an original dependency treebank rather than a constituent treebank, - or at least linguistically revised at the dependency level. Making the constituent-dependency conversion principles (Johansson & Nugues 2007, forthcoming) public *before* rather than after the shared task might also have contributed to a better CG annotation transfer.

3 System architecture

As described in (Bick 2006), the LingPars system uses the fine-grained part of speech (PoS) tags (POSTAG) and - for words above a certain frequency threshold - the LEMMA or, if absent, FORM tag. In a first round, LingPars calculates a preference list of functions and dependencies for each word, examining all possible mother-daughter pairs and n-grams in the sentence (or paragraph). Next, dependencies are adjusted for function, basically summing up the frequency-, distance- and direction-calibrated function->PoS attachment probabilities for all contextually allowed functions for a given word. Finally, dependency probabilities are weighted using linked probabilities for possible mother-, daughter- and sister-tags in a second pass.

The result are 2 arrays, one for possible daughter->mother pairs, one for word:function pairs. LingPars then attempts to "effectuate" the dependency (daughter->mother) array, starting with the - in normalized terms - highest value. If the daughter candidate is as yet unattached, and the dependency does not produce circularities or crossing branches, the corresponding part of the (ordered) word:function array is calibrated for the suggested dependency, and the top-ranking function chosen.

One of the major problems in the original system was uniqueness clashes, and as a special case, root attachment ambiguity, resulting from a conflict between the current best attachment candidate in the pipe and an earlier chosen attachment to the same head. Originally, the parser tried to resolve these conflicts by assigning penalties to the attachments in question and recalculating "second best" attachments for the tokens in question. While solving some cases, this method often timed out without finding a globally compatible solution.

In the new version of LingPars, with open resources, the attachment and function label rankings were calibrated using the analysis suggested by the EngGram CG system for the same data, assigning extra weights to readings supported by the rule based analysis, using addition of a weight constant for function, and multiplication with a weight constant for attachments, thus integrating CG information on par with statistical information[1]. This was

[1] Experiments suggested that there is a limit beyond which an increase of these weighting constants, for both function and dependency, will actually lead to a *decrease* in performance, because the positive effect of long-distance attachments from the CG system will be cancelled out by the negative effect of

not, however, thought sufficient to resolve the global syntactic problem of root attachment where (wrong) statistical preferences could be so strong that even 20 rounds of penalties could not weaken them sufficient to be ruled out. Therefore, root and root attachments supported by the CG trees were fixed in the first pass, without reruns. The same method was used for another source of global errors - coordination. Here, the probabilistic system had difficulties learning patterns, because a specific function label (SBJ or OBJ etc) would be associated with a non-specific word class (CC), and a non-specific function (COORD) with a host of different word classes. Again, adding a first-pass override based on CG-provided coordination links solved many of these cases.

Though limited to 2 types of global dependency (root and coordination), the help provided by the rule based analysis, also had indirect benefits by providing a better point of departure for other attachments, among other things because LingPars exaggerated both good and bad analyses: Good attachments would help weight other attachments through correct n-gram-, mother-, daughter- and sibling contexts, but isolated bad attachments would lead to even worse attachments by triggering, for instance, incorrect BARRIER or crossing branch constraints. These adverse effects were moderated by getting a larger percentage of global dependencies right in the first place, and also by a new addition to the crossing and BARRIER subroutine invalidating it in the case of CG-supported attachments.

4 Evaluation

The hybrid LingPars was the best-scoring system in the open section of both domain adaptation tasks[2] (Nivre et al. 2007), outperforming its probabilistic core system on all scores, with an improvement of 6.57 LAS percentage points for the pchemtb corpus (table 1), and 3.42 for the CHILDES attachment score (table 2). In the former, the effect was slightly more marked for attachment than for label accuracy.

However, whereas results also surpassed those of the top *closed class* system in the CHILDES domain (by 1.12 percentage points), they fell short of this mark for the pchemtb corpus - by 1.26 percentage points for label accuracy and 1.80 for attachment.

	Top score pchemtb	***average pchemtb***	***System pchemtb***	***System train***
Closed				
LAS	81.06	73.03	71.81	(75.01)[3]
UAS	83.42	76.42	74.71	(76.71)
LS	88.28	81.74	80.78	(84.12)
Open				
LAS	78.48	65.11	78.48	(79.04)
UAS	81.62	70.24	81.62	(80.82)
LS	87.02	77.14	87.02	(88.07)

Table 1: Performance, Pchemtb data

UAS	***Top score***	***average***	***System***
CHILDES closed	61.37	57.89	58.07
CHILDES open	62.49	56.12	62.49

Table 2: Performance, CHILDES data

When compared with runs on (unknown) data from the training domain, cross-domain performance of the closed system was 2 percentage points lower for attachment and 3.5 lower for label accuracy (LA scores of 71.81 and 58.07 for the pchemtb and CHILDES corpus, respectively).

Interestingly, hybrid results for the pchemtb data were only marginally lower than for the training domain (in fact, *higher* for attachment), suggesting a higher domain robustness for the hybrid than for the probabilistic approach.

disturbing the application of machine-learned local dependencies.

[2] During the test phase, the data set for one of the originally 2 test domains, CHILDES, was withdrawn from the official ranking, though its scores were still computed and admissible for evaluation.

[3]This is the accuracy for the test data used during development. For the PTB gold test data from track 1, LAS was higher (76.21).

References

E. Bick. 2006, LingPars, a Linguistically Inspired, Language-Independent Machine Learner for Dependency Treebanks, In: Màrquez, Lluís & Klein, Dan (eds.), *Proceedings of the Tenth Conference on Natural Language Learning (CoNLL-X, New York, June 8-9, 2006)*

E. Bick. 2005. Turning Constraint Grammar Data into Running Dependency Treebanks. In: Civit, Montserrat & Kübler, Sandra & Martí, Ma. Antònia (red.), *Proceedings of TLT 2005, Barcelona.* pp.19-2

R. Brown. 1973. A First Language: The Early Stages. Harvard University Press

R. Johansson and P. Nugues. 2007. Extended Constituent-to-Dependency Conversion for English. In: *Proceedings of NoDaLiDa 16.* Forthcoming

F. Karlsson, A. Vouitilainen, J. Heikkilä and A. Anttila. 1995. Constraint Grammar - A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter: Berlin.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein and L. Ungar. 2004. Integrated Annotation for biomedical Information Extractions. In: *Proceedings of HLT-NAACL 2004.*

B. MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Lawrence Erlbaum

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. In: *Computational Linguistics Vol. 19,2.* pp. 313-330

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel and D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In: *Proceedings of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

A Constraint Satisfaction Approach to Dependency Parsing

Sander Canisius
ILK / Communication and Information Science
Tilburg University, P.O. Box 90153,
NL-5000 LE Tilburg, The Netherlands
S.V.M.Canisius@uvt.nl

Erik Tjong Kim Sang
ISLA, University of Amsterdam,
Kruislaan 403, NL-1098 SJ Amsterdam,
The Netherlands
erikt@science.uva.nl

Abstract

We present an adaptation of constraint satisfaction inference (Canisius et al., 2006b) for predicting dependency trees. Three different classifiers are trained to predict weighted soft-constraints on parts of the complex output. From these constraints, a standard weighted constraint satisfaction problem can be formed, the solution to which is a valid dependency tree.

1 Introduction

Like the CoNLL-2006 shared task, the 2007 shared task focuses on dependency parsing and aims at comparing state-of-the-art machine learning algorithms applied to this task (Nivre et al., 2007). For our official submission, we used the dependency parser described by Canisius et al. (2006a). In this paper, we present a novel approach to dependency parsing based on constraint satisfaction. The method is an adaptation of earlier work using constraint satisfaction techniques for predicting sequential outputs (Canisius et al., 2006b). We evaluated our approach on all ten data sets of the 2007 shared task[1].

In the remainder of this paper, we will present the new constraint satisfaction method for dependency parsing in Section 2. The method is evaluated in Section 3, in which we will also present a brief error analysis. Finally, Section 4 presents our main conclusions.

2 Constraint Satisfaction Inference for Dependency Trees

The parsing algorithm we used is an adaptation for dependency trees of the constraint satisfaction inference method for sequential output structures proposed by Canisius et al. (2006b). The technique uses standard classifiers to predict a weighted constraint satisfaction problem, the solution to which is the complete dependency tree. Constraints that are predicted each cover a small part of the complete tree, and overlap between them ensures that global output structure is taken into account, even though the classifiers only make local predictions in isolation of each other.

A weighted constraint satisfaction problem (W-CSP) is a tuple (X, D, C, W). Here, $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables. $D(x)$ is a function that maps each variable to its domain, and C is a set of constraints on the values assigned to the variables. For a traditional (non-weighted) constraint satisfaction problem, a valid solution is an assignment of values to the variables that (1) are a member of the corresponding variable's domain, and (2) satisfy *all* constraints in the set C. Weighted constraint satisfaction, however, relaxes this requirement to satisfy all constraints. Instead, constraints are assigned weights that may be interpreted as reflecting the importance of satisfying that constraint. The optimal solution to a W-CSP is the solution that assigns those values that maximise the sum of the weights of satisfied constraints.

[1] Hajič et al. (2004), Aduriz et al. (2003), Martí et al. (2007), Chen et al. (2003), Böhmová et al. (2003), Marcus et al. (1993), Johansson and Nugues (2007), Prokopidis et al. (2005), Csendes et al. (2005), Montemagni et al. (2003), Oflazer et al. (2003)

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1124–1128,
Prague, June 2007.

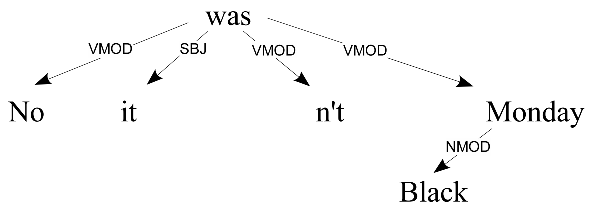


Figure 1: Dependency tree for the sentence *No it wasn't Black Monday*

To adapt this framework to predicting a dependency tree for a sentence, we construct a constraint satisfaction problem by first introducing one variable x_i for each token of the sentence. This variable's value corresponds to the dependency relation that token is the modifier of, i.e. it should specify a relation type and a head token. The constraints of the CSP are predicted by a classifier, where the weight for a constraint corresponds to the classifier's confidence estimate for the prediction.

For the current study, we trained three classifiers to predict three different types of constraints.

1. $C_{dep}(\textit{head}, \textit{modifier}, \textit{relation})$, i.e. the resulting dependency tree should have a dependency arc from *head* to *modifier* labelled with type *relation*. For the example tree in Figure 1, among others the constraint $C_{dep}(\textit{head=was}, \textit{modifier=No}, \textit{relation=VMOD})$ should be predicted.

2. $C_{dir}(\textit{modifier}, \textit{direction})$, the relative position (i.e. to its left or to its right) of the head of *modifier*. The tree in Figure 1 will give rise to constraints such as $C_{dir}(\textit{modifier=Black}, \textit{direction=RIGHT})$.

3. $C_{mod}(\textit{head}, \textit{relation})$, in the dependency tree, *head* should be modified by a relation of type *relation*. The constraints generated for the word *was* in Figure 1 would be $C_{mod}(\textit{head=was}, \textit{relations=SBJ})$, and $C_{mod}(\textit{head=was}, \textit{relations=VMOD})$.

Predicting constraints of type C_{dep} is essentially what is done by Canisius et al. (2006a); a classifier is trained to predict a relation label, or a symbol signalling the absence of a relation, for each pair of tokens in a sentence[2]. The training data for this classifier consists of positive examples of constraints to generate, e.g. $\textit{was}, \textit{No}, \textit{VMOD}$, and negative examples, of constraints *not* to generate, e.g. $\textit{was}, \textit{Black}, \textit{NONE}$, but also $\textit{No}, \textit{was}, \textit{NONE}$. In the aforementioned paper, it is shown that downsampling the negative class in the classifier's training data improves the recall for predicted constraints. The fact that improved recall comes at the cost of a reduced precision is compensated for by our choice for the weighted constraint satisfaction framework: an overpredicted constraint may still be left unsatisfied if other, conflicting constraints outweigh its own weight.

In addition to giving rise to a set of constraints, this classifier differs from the other two in the sense that it is also used to predict the domains of the variables, i.e. any dependency relation not predicted by this classifier will not be considered for inclusion in the output tree.

Whereas the C_{dep} classifier classifies instances for each pair of words, the classifiers for C_{dir} and C_{mod} only classify individual tokens. The features for these classifiers have been kept simple and the same for both classifiers: a 5-slot wide window of both tokens and part-of-speech tags, centred on the token currently being classified. The two classifiers differ in the classes they predict. For C_{dir}, there are only three possible classes: `LEFT`, `RIGHT`, `NONE`. Instances classified as `LEFT`, or `RIGHT` give rise to constraints, whereas `NONE` implies that no C_{dir} constraint is added for that token.

For C_{mod} there is a rather large class space; a class label reflects all modifying relations for the token, e.g. `SBJ+VMOD`. From this label, as many constraints are generated as there are different relation types in the label.

With the above, a weighted constraint satisfaction problem can be formulated that, when solved, describes a dependency tree. As we formulated our problem as a constraint satisfaction problem, any off-the-shelf W-CSP solver could be used to obtain the best dependency parse. However, in general such solvers have a time complexity exponential in the

[2] For reasons of efficiency and to avoid having too many negative instances in the training data, we follow the approach of Canisius et al. (2006a) of limiting the maximum distance between a potential head and modifier.

Language	LAS	’06	UAS	’06
Arabic	60.36	+1.2	78.61	+1.7
Basque	64.23	+1.1	72.24	+2.1
Catalan	77.33	+1.9	84.73	+3.1
Chinese	71.73	+1.3	77.29	+2.5
Czech	57.58	+1.4	75.61	+3.5
English	79.47	+2.2	81.05	+2.8
Greek	62.32	+2.0	76.42	+4.0
Hungarian	66.86	+2.6	72.52	+4.7
Italian	77.04	+1.5	81.24	+2.2
Turkish	67.80	-0.3	75.58	+0.4

Table 1: Performance of the system applied to the test data for each language. The ’06 columns show the gain/loss with respect to the parser of Canisius et al. (2006a).

Language	’06	C_{dep}	$C_{dep}^{mod/}$	$C_{dep}^{dir/}$	all
Arabic	59.13	+0.3	+0.9	+0.9	+1.2
Basque	63.17	+0.3	+0.4	+0.9	+1.1
Catalan	75.44	+0.8	+1.2	+1.4	+1.9
Chinese	70.45	+0.4	+1.2	+0.4	+1.3
Czech	56.14	+0.5	+0.5	+1.1	+1.4
English	77.27	+0.4	+1.4	+1.2	+2.2
Greek	60.35	+0.4	+0.6	+1.6	+2.0
Hungarian	64.31	+1.9	+1.3	+2.8	+2.6
Italian	75.57	+0.2	+1.0	+1.1	+1.5
Turkish	68.09	-0.2	-0.3	-0.3	-0.3

Table 2: Performance of the parser by Canisius et al. (2006a) and the performance gain of the constraint satisfaction inference parser with various constraint configurations.

number of variables, and thus in the length of the sentence. As a more efficient alternative we chose to use the CKY algorithm for dependency parsing (Eisner, 2000) for computing the best solution, which has only cubic time complexity, but comes with the disadvantage of only considering projective trees as candidate solutions.

3 Results and discussion

We tested our system on all ten languages of the shared task. The three constraint classifiers have been implemented with memory-based learning. No language-specific parameter optimisation or feature engineering has been performed, but rather the exact same system has been applied to all languages. Labelled and unlabelled attachment scores are listed in Table 1. In addition, we show the increase/decrease in performance when compared with the parser of Canisius et al. (2006a); for all languages but Turkish, there is a consistent increase, mostly somewhere between 1.0 and 2.0 percent in labelled attachment score.

The parser by Canisius et al. (2006a) can be considered a rudimentary implementation of constraint satisfaction inference that only uses C_{dep} constraints. The parser described in this paper elaborates this by adding (1) the C_{mod} and C_{dir} *soft* constraints, and (2) projectivity and acyclicity *hard* constraints, enforced implicitly by the CKY algorithm.

To evaluate the effect of each of these constraints, Table 2 shows the labelled attachment scores for several parser configurations; starting with the 2006 parser, i.e. a parser with only C_{dep} constraints, then the CKY-driven C_{dep} parser, i.e. with acyclicity and projectivity constraints, then with C_{mod}, and C_{dir} separately, and finally, the full parser based on all constraints. It can be seen that supplementing the C_{dep}-only parser with hard constraints for acyclicity and projectivity already gives a small performance improvement. For some languages, such as Italian (+0.2), this improvement is rather small, however for Hungarian 1.9 is gained only by using CKY. The remaining columns show that adding more constraints improves performance, and that for all languages but Turkish and Hungarian, using all constraints works best.

While in comparison with the system of Canisius et al. (2006a) the addition of extra constraints has clearly shown its use, we expect the C_{dep} classifier still to be the performance bottleneck of the system. This is mainly due to the fact that this classifier is also responsible for defining the domains of the CSP variables, i.e. which dependency relations will be considered for inclusion in the output. For this reason, we performed an error analysis of the output of the C_{dep} classifier and the effect it has on the performance of the complete system.

In our error analysis, we distinguish three types of errors: 1) *label errors*, a correct dependency arc was added to the tree, but its label is incorrect, 2) *recall*

	C_{dep}		prec.	rec.
Language	**prec.**	**rec.**	**%OOD**	**%OOD**
Arabic	54.90	73.66	78.83	77.95
Basque	55.82	74.10	85.05	83.66
Catalan	65.19	87.25	80.29	80.00
Chinese	65.10	76.49	83.79	82.94
Czech	53.64	74.35	81.16	80.27
English	59.37	90.08	67.51	66.63
Greek	53.24	76.29	79.96	79.08
Hungarian	44.71	78.64	69.08	67.45
Italian	71.70	82.57	87.97	87.32
Turkish	64.92	72.79	89.11	88.51

Table 3: Columns two and three: precision and recall on dependency predictions by the C_{dep} classifier. Columns four and five: percentage of dependency arc precision and recall errors caused by out-of-domain errors.

errors, the true dependency tree contains an arc that is missing from the predicted tree, and 3) *precision errors*, the predicted tree contains a dependency arc that is not part of the true dependency parse.

Label errors are always a direct consequence of erroneous C_{dep} predictions. If the correct arc was predicted, but with an incorrect label, then by definition, the correct arc with the correct label cannot have been predicted at the same time. In case of the other two types of errors, the correct constraints may well have been predicted, but afterwards outweighed by other, conflicting constraints. Nevertheless, precision and recall errors may also be caused by the fact that the C_{dep} classifier simply did not predict a dependency arc where it should have. We will refer to those errors as out-of-domain errors, since the domain of at least one of the CSP variables does not contain the correct value. An out-of-domain error is a direct consequence of a recall error made by the C_{dep} classifier. To illustrate these interactions, Table 3 shows for all languages the precision and recall of the C_{dep} classifier, and the percentage of dependency precision and recall errors that are out-of-domain errors.

The table reveals several interesting facts. For English, which is the language for which our system attains its highest score, the percentage of dependency precision and recall errors caused by C_{dep} recall errors is the lowest of all languages. This can directly be related to the 90% recall of the English C_{dep} classifier. Apparently, the weak precision (59%), caused by down-sampling the training data, is compensated for in the subsequent constraint satisfaction process.

For Italian, the percentage of out-of-domain-related errors is much higher than for English. At the same time, the precision and recall of the C_{dep} classifier are much more in balance, i.e. a higher precision, but a lower recall. We tried breaking this balance in favour of a higher recall by applying an even stronger down-sampling of negative instances, and indeed the parser benefits from this. Labelled attachment increases from 77.04% to 78.41%. The precision and recall of this new C_{dep} classifier are 58.65% and 87.15%, respectively.

The lowest C_{dep} precision has been observed for Hungarian (44.71), which unfortunately is not mirrored by a high recall score. Remarkably however, after English, Hungarian has the lowest percentage of dependency errors due to C_{dep} recall errors (69.08 and 67.45). It is therefore hypothesised that not the low recall, but the low precision is the main cause for errors made on Hungarian. With this in mind, we briefly experimented with weaker down-sampling ratios in order to boost precision, but so far we did not manage to attain better results.

4 Concluding remarks

We have presented a novel dependency parsing method based on a standard constraint satisfaction framework. First results on a set of ten different languages have been promising, but so far no extensive optimisation has been performed, which inevitably reflects upon the scores attained by the system. Future work will focus on tuning the many parameters our system has, as well as on experimenting with different types of constraints to supplement or replace one or more of the three types used in this study.

Acknowledgements

The authors wish to thank Antal van den Bosch for discussions and suggestions. This research is funded by NWO, the Netherlands Organization for Scientific Research under the IMIX programme.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

S. Canisius, T. Bogers, A. van den Bosch, J. Geertzen, and E. Tjong Kim Sang. 2006a. Dependency parsing by inference over high-recall dependency predictions. In *Proceedings of CoNLL-X*. New York, NY, USA.

S. Canisius, A. van den Bosch, and W. Daelemans. 2006b. Constraint Satisfaction Inference: Non-probabilistic Global Inference for Sequence Labelling. *Proceedings of the EACL 2006 Workshop on Learning Structured Information in Natural Language Applications*, pages 9–16.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

A Two-stage Parser for Multilingual Dependency Parsing

Wenliang Chen, Yujie Zhang, Hitoshi Isahara
Computational Linguistics Group
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
{chenwl, yujie, isahara}@nict.go.jp

Abstract

We present a two-stage multilingual dependency parsing system submitted to the Multilingual Track of CoNLL-2007. The parser first identifies dependencies using a deterministic parsing method and then labels those dependencies as a sequence labeling problem. We describe the features used in each stage. For four languages with different values of ROOT, we design some special features for the ROOT labeler. Then we present evaluation results and error analyses focusing on Chinese.

1 Introduction

The CoNLL-2007 shared tasks include two tracks: the Multilingual Track and Domain Adaptation Track(Nivre et al., 2007). We took part the Multilingual Track of all ten languages provided by the CoNLL-2007 shared task organizers(Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003) .

In this paper, we describe a two-stage parsing system consisting of an unlabeled parser and a sequence labeler, which was submitted to the Multilingual Track. At the first stage, we use the parsing model proposed by (Nivre, 2003) to assign the arcs between the words. Then we obtain a dependency parsing tree based on the arcs. At the second stage, we use a SVM-based approach(Kudo and Matsumoto, 2001) to tag the dependency label for each arc. The labeling is treated as a sequence labeling problem. We design some special features for tagging the labels of ROOT for Arabic, Basque, Czech, and Greek, which have different labels for ROOT. The experimental results show that our approach can provide higher scores than average.

2 Two-Stage Parsing

2.1 The Unlabeled Parser

The unlabeled parser predicts unlabeled directed dependencies. This parser is primarily based on the parsing models described by (Nivre, 2003). The algorithm makes a dependency parsing tree in one left-to-right pass over the input, and uses a stack to store the processed tokens. The behaviors of the parser are defined by four elementary actions (where TOP is the token on top of the stack and NEXT is the next token in the original input string):

- Left-Arc(LA): Add an arc from NEXT to TOP; pop the stack.
- Right-Arc(RA): Add an arc from TOP to NEXT; push NEXT onto the stack.
- Reduce(RE): Pop the stack.
- Shift(SH): Push NEXT onto the stack.

Although (Nivre et al., 2006) used the pseudo-projective approach to process non-projective dependencies, here we only derive projective dependency tree. We use MaltParser(Nivre et al., 2006)

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1129–1133,
Prague, June 2007.

V0.4[1] to implement the unlabeled parser, and use the SVM model as the classifier. More specifically, the MaltParser use LIBSVM(Chang and Lin, 2001) with a quadratic kernel and the built-in one-versus-all strategy for multi-class classification.

2.1.1 Features for Parsing

The MaltParser is a history-based parsing model, which relies on features of the derivation history to predict the next parser action. We represent the features extracted from the fields of the data representation, including FORM, LEMMA, CPOSTAG, POSTAG, and FEATS. We use the features for all languages that are listed as follows:

- The FORM features: the FORM of TOP and NEXT, the FORM of the token immediately before NEXT in original input string, and the FORM of the head of TOP.
- The LEMMA features: the LEMMA of TOP and NEXT, the LEMMA of the token immediately before NEXT in original input string, and the LEMMA of the head of TOP.
- The CPOS features: the CPOSTAG of TOP and NEXT, and the CPOSTAG of next left token of the head of TOP.
- The POS features: the POSTAG of TOP and NEXT, the POSTAG of next three tokens after NEXT, the POSTAG of the token immediately before NEXT in original input string, the POSTAG of the token immediately below TOP, and the POSTAG of the token immediately after rightmost dependent of TOP.
- The FEATS features: the FEATS of TOP and NEXT.

But note that the fields LEMMA and FEATS are not available for all languages.

2.2 The Sequence Labeler

2.2.1 The Sequence Problem

We denote by $x = x_1, ..., x_n$ a sentence with n words and by y a corresponding dependency tree. A dependency tree is represented from ROOT to leaves with a set of ordered pairs $(i, j) \in y$ in which x_j is a dependent and x_i is the head. We have produced the dependency tree y at the first stage. In this stage, we assign a label $l_{(i,j)}$ to each pair.

As described in (McDonald et al., 2006), we treat the labeling of dependencies as a sequence labeling problem. Suppose that we consider a head x_i with dependents $x_{j1}, ..., x_{jM}$. We then consider the labels of $(i, j1), ..., (i, jM)$ as a sequence. We use the model to find the solution:

$$l_{max} = \arg\max_{l} s(l, i, y, x) \qquad (1)$$

And we consider a first-order Markov chain of labels.

We used the package YamCha (V0.33)[2] to implement the SVM model for labeling. YamCha is a powerful tool for sequence labeling(Kudo and Matsumoto, 2001).

2.2.2 Features for Labeling

After the first stage, we know the unlabeled dependency parsing tree for the input sentence. This information forms the basis for part of the features of the second stage. For the sequence labeler, we define the individual features, the pair features, the verb features, the neighbor features, and the position features. All the features are listed as follows:

- The individual features: the FORM, the LEMMA, the CPOSTAG, the POSTAG, and the FEATS of the parent and child node.
- The pair features: the direction of dependency, the combination of lemmata of the parent and child node, the combination of parent's LEMMA and child's CPOSTAG, the combination of parent's CPOSTAG and child's LEMMA, and the combination of FEATS of parent and child.
- The verb features: whether the parent or child is the first or last verb in the sentence.
- The neighbor features: the combination of CPOSTAG and LEMMA of the left and right neighbors of the parent and child, number of children, CPOSTAG sequence of children.

[1] The tool is available at
http://w3.msi.vxu.se/˜nivre/research/MaltParser.html

[2] YamCha is available at
http://chasen.org/˜taku/software/yamcha/

- The position features: whether the child is the first or last word in the sentence and whether the child is the first word of left or right of parent.

2.2.3 Features for the Root Labeler

Because there are four languages have different labels for root, we define the features for the root labeler. The features are listed as follows:

- The individual features: the FORM, the LEMMA, the CPOSTAG, the POSTAG, and the FEATS of the parent and child node.

- The verb features: whether the child is the first or last verb in the sentence.

- The neighbor features: the combination of CPOSTAG and LEMMA of the left and right neighbors of the parent and child, number of children, CPOSTAG sequence of children.

- The position features: whether the child is the first or last word in the sentence and whether the child is the first word of left or right of parent.

3 Evaluation Results

We evaluated our system in the Multilingual Track for all languages. For the unlabeled parser, we chose the parameters for the MaltParser based on performance from a held-out section of the training data. We also chose the parameters for Yamcha based on performance from training data.

Our official results are shown at Table 1. Performance is measured by labeled accuracy and unlabeled accuracy. These results showed that our two-stage system can achieve good performance. For all languages, our system provided better results than average performance of all the systems(Nivre et al., 2007). Compared with top 3 scores, our system provided slightly worse performance. The reasons may be that we just used projective parsing algorithms while all languages except Chinese have non-projective structure. Another reason was that we did not tune good parameters for the system due to lack of time.

Data Set	LA	UA
Arabic	74.65	83.49
Basque	72.39	78.63
Catalan	86.66	90.87
Chinese	81.24	85.91
Czech	73.69	80.14
English	83.81	84.91
Greek	74.42	81.16
Hungarian	75.34	79.25
Italian	82.04	85.91
Turkish	76.31	81.92
average	78.06	83.22

Table 1: The results of proposed approach. LABELED ATTACHMENT SCORE(LA) and UNLABELED ATTACHMENT SCORE(UA)

4 General Error Analysis

4.1 Chinese

For Chinese, the system achieved 81.24% on labeled accuracy and 85.91% on unlabeled accuracy. We also ran the MaltParser to provide the labels. Besides the same features, we added the DEPREL features: the dependency type of TOP, the dependency type of the token leftmost of TOP, the dependency type of the token rightmost of TOP, and the dependency type of the token leftmost of NEXT. The labeled accuracy of MaltParser was 80.84%, 0.4% lower than our system.

Some conjunctions, prepositions, and DE[3] attached to their head words with much lower accuracy: 74% for DE, 76% for conjunctions, and 71% for prepositions. In the test data, these words formed 19.7%. For Chinese parsing, coordination and preposition phrase attachment were hard problems. (Chen et al., 2006) defined the special features for coordinations for chunking. In the future, we plan to define some special features for these words.

Now we focused words where most of the errors occur as Table 2 shows. For “的/DE”, there was 32.4% error rate of 383 occurrences. And most of them were assigned incorrect labels between “property” and “predication”: 45 times for “property” instead of “predication” and 20 times for “predication” instead of “property”. For examples, “的/DE”

[3] including “的/得/地/之”.

	num	any	head	dep	both
的/ DE	383	124	35	116	27
、/ C	117	38	36	37	35
在/ P	67	20	6	19	5
台灣/ N	31	10	8	4	2
是/ V	72	8	8	8	8

Table 2: The words where most of errors occur in Chinese data.

in “流行/的/電視/頻道(popular TV channel)” was to be tagged as “property” instead of “predication”, while “的/DE” in “博物館/的/志工(volunteer of museum)” was to be tagged as “predication” instead of “property”. It was very hard to tell the labels between the words around “的”. Humans can make the distinction between property and predication for “的”, because we have background knowledge of the words. So if we can incorporate the additional knowledge for the system, the system may assign the correct label.

For “、/C”, it was hard to assign the head, 36 wrong head of all 38 errors. It often appeared at coordination expressions. For example, the head of “、” at “在/酷/斃/了/、/太/炫/了/之外/(Besides extreme cool and too amazing)” was “之外”, and the head of “、” at “提供給/參觀/者/深厚/、/有/系統/的/知識(Give the visitors solid and methodical knowledge)” was “知識”.

5 Conclusion

In this paper, we presented our two-stage dependency parsing system submitted to the Multilingual Track of CoNLL-2007 shared task. We used Nivre’s method to produce the dependency arcs and the sequence labeler to produce the dependency labels. The experimental results showed that our system can provide good performance for all languages.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

C.C. Chang and C.J. Lin. 2001. LIBSVM: a library for support vector machines. *Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm*, 80:604–611.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of chinese chunking. In *COLING/ACL 2006(Poster Sessions)*, Sydney, Australia, July.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *In Proceedings of NAACL01*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City, June. Association for Computational Linguistics.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

Incremental Dependency Parsing Using Online Learning

Richard Johansson and **Pierre Nugues**
Department of Computer Science, Lund University, Sweden
{richard, pierre}@cs.lth.se

Abstract

We describe an incremental parser that was trained to minimize cost over sentences rather than over individual parsing actions. This is an attempt to use the advantages of the two top-scoring systems in the CoNLL-X shared task.

In the evaluation, we present the performance of the parser in the Multilingual task, as well as an evaluation of the contribution of bidirectional parsing and beam search to the parsing performance.

1 Introduction

The two best-performing systems in the CoNLL-X shared task (Buchholz and Marsi, 2006) can be classified along two lines depending on the method they used to train the parsing models. Although the parsers are quite different, their creators could report near-tie scores. The approach of the top system (McDonald et al., 2006) was to fit the model to minimize cost over sentences, while the second-best system (Nivre et al., 2006) trained the model to maximize performance over individual decisions in an incremental algorithm. This difference is a natural consequence of their respective parsing strategies: CKY-style maximization of link score and incremental parsing.

In this paper, we describe an attempt to unify the two approaches: an incremental parsing strategy that is trained to maximize performance over sentences rather than over individual parsing actions.

2 Parsing Method

2.1 Nivre's Parser

We used Nivre's algorithm (Nivre et al., 2006), which is a variant of the shift–reduce parser. Like the regular shift–reduce, it uses a stack S and a list of input words W, and builds the parse tree incrementally using a set of parsing actions (see Table 1). It can be shown that Nivre's parser creates projective and acyclic graphs and that every projective dependency graph can be produced by a sequence of parser actions. In addition, the worst-case number of actions is linear with respect to the number of words in the sentence.

2.2 Handling Nonprojective Parse Trees

While the parsing algorithm produces projective trees only, nonprojective arcs can be handled using a preprocessing step before training the model and a postprocessing step after parsing the sentences.

The projectivization algorithm (Nivre and Nilsson, 2005) iteratively moves each nonprojective arc upward in the tree until the whole tree is projective. To be able to recover the nonprojective arcs after parsing, the projectivization operation replaces the labels of the arcs it modifies with traces indicating which links should be moved and where attach to attach them (the "Head+Path" encoding). The model is trained with these new labels that makes it possible to carry out the reverse operation and produce nonprojective structures.

2.3 Bidirectional Parsing

Shift-reduce is by construction a directional parser, typically applied from left to right. To make better use of the training set, we applied the algorithm in both directions as Johansson and Nugues (2006) and Sagae and Lavie (2006) for all languages except Catalan and Hungarian. This, we believe, also has the advantage of making the parser less sensitive to whether the language is head-initial or head-final.

We trained the model on projectivized graphs from left to right and right to left and used a voting strategy based on link scores. Each link was assigned a score (simply by using the score of the `la` or `ra` actions for each link). To resolve the conflicts

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1134–1138,
Prague, June 2007. ©2007 Association for Computational Linguistics

Table 1: Nivre's parser transitions where W is the initial word list; I, the current input word list; A, the graph of dependencies; and S, the stack. (n', n) denotes a dependency relations between n' and n, where n' is the head and n the dependent.

Actions	**Parser actions**	**Conditions**
Initialize	$\langle nil, W, \emptyset \rangle$	
Terminate	$\langle S, nil, A \rangle$	
Left-arc	$\langle n \mid S, n' \mid I, A \rangle \rightarrow \langle S, n' \mid I, A \cup \{(n', n)\} \rangle$	$\neg \exists n''(n'', n) \in A$
Right-arc	$\langle n \mid S, n' \mid I, A \rangle \rightarrow \langle n' \mid n \mid S, I, A \cup \{(n, n')\} \rangle$	$\neg \exists n''(n'', n') \in A$
Reduce	$\langle n \mid S, I, A \rangle \rightarrow \langle S, I, A \rangle$	$\exists n'(n', n) \in A$
Shift	$\langle S, n \mid I, A \rangle \rightarrow \langle n \mid S, I, A \rangle$	

between the two parses in a manner that makes the tree projective, single-head, rooted, and cycle-free, we applied the Eisner algorithm (Eisner, 1996).

2.4 Beam Search

As in our previous parser (Johansson and Nugues, 2006), we used a beam-search extension to Nivre's original algorithm (which is greedy in its original formulation). Each parsing action was assigned a score, and the beam search allows us to find a better overall score of the sequence of actions. In this work, we used a beam width of 8 for Catalan, Chinese, Czech, and English and 16 for the other languages.

3 Learning Method

3.1 Overview

We model the parsing problem for a sentence $\boldsymbol{x}$ as finding the parse $\hat{y} = \arg\max_y F(\boldsymbol{x}, y)$ that maximizes a discriminant function F. In this work, we consider linear discriminants of the following form:

$$F(\boldsymbol{x}, y) = \boldsymbol{w} \cdot \boldsymbol{\Psi}(\boldsymbol{x}, y)$$

where $\boldsymbol{\Psi}(\boldsymbol{x}, y)$ is a numeric feature representation of the pair $(\boldsymbol{x}, y)$ and $\boldsymbol{w}$ a vector of feature weights. Learning F in this case comes down to assigning good weights in the vector $\boldsymbol{w}$.

Machine learning research for similar problems have generally used margin-based formulations. These include global batch methods such as *SVM^{struct}* (Tsochantaridis et al., 2005) as well as online methods such as the Online Passive-Aggressive Algorithm (OPA) (Crammer et al., 2006). Although the batch methods are formulated very elegantly, they do not seem to scale well to the large training sets prevalent in NLP contexts – we briefly considered using *SVM^{struct}* but training was too time-consuming. The online methods on the other hand, although less theoretically appealing, can handle realistically sized data sets and have successfully been applied in dependency parsing (McDonald et al., 2006). Because of this, we used the OPA algorithm throughout this work.

3.2 Implementation

In the online learning framework, the weight vector is constructed incrementally. At each step, it computes an update to the weight vector based on the current example. The resulting weight vector is frequently overfit to the last examples. One way to reduce overfitting is to use the average of all successive weight vectors as the result of the training (Freund and Schapire, 1999).

Algorithm 1 shows the algorithm. It uses an "aggressiveness" parameter C to reduce overfitting, analogous to the C parameter in SVMs. The algorithm also needs a cost function ρ, which describes how much a parse tree deviates from the gold standard. In this work, we defined ρ as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link. The number of iterations was 5 for all languages.

For a sentence $\boldsymbol{x}$ and a parse tree y, we defined the feature representation by finding the sequence $\langle \langle S_1, I_1 \rangle, a_1 \rangle, \langle \langle S_2, I_2 \rangle, a_2 \rangle \ldots$ of states and their corresponding actions, and creating a feature vector for each state/action pair. The discriminant function was thus written

$$\boldsymbol{\Psi}(\boldsymbol{x}, y) \cdot \boldsymbol{w} = \sum_i \psi(\langle S_i, I_i \rangle, a_i) \cdot \boldsymbol{w}$$

where ψ is a feature function that assigns a feature

Algorithm 1 The Online PA Algorithm

```
input Training set $\mathcal{T} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
      Number of iterations $N$
      Regularization parameter $C$
      Cost function $\rho$
Initialize $w$ to zeros
repeat $N$ times
  for $(\boldsymbol{x}_t, y_t)$ in $\mathcal{T}$
    let $\tilde{y}_t = \arg\max_y F(\boldsymbol{x}_t, y) + \sqrt{\rho(y_t, y)}$
    let $\tau_t = \min\left(C, \frac{F(\boldsymbol{x}_t,\tilde{y}_t) - F(\boldsymbol{x}_t,y_t) + \sqrt{\rho(y_t,\tilde{y}_t)}}{\|\boldsymbol{\Psi}(\boldsymbol{x},y_t) - \boldsymbol{\Psi}(\boldsymbol{x},\tilde{y}_t)\|^2}\right)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \tau_t(\boldsymbol{\Psi}(\boldsymbol{x}, y_t) - \boldsymbol{\Psi}(\boldsymbol{x}, \tilde{y}_t))$
return $\boldsymbol{w}_{\text{average}}$
```

vector to a state $\langle S_i, I_i \rangle$ and the action a_i taken in that state. Table 2 shows the feature sets used in ψ for all languages. In principle, a kernel could also be used, but that would degrade performance severely. Instead, we formed a new vector by combining features pairwisely – this is equivalent to using a quadratic kernel.

Since the history-based feature set used in the parsing algorithm makes it impossible to use independence to factorize the scoring function, an exact search to find the best-scoring action sequence ($\arg\max_y$ in Algorithm 1) is not possible. However, the beam search allows us to find a reasonable approximation.

4 Results

Table 3 shows the results of our system in the Multilingual task.

4.1 Compared to SVM-based Local Classifiers

We compared the performance of the parser with a parser based on local SVM classifiers (Johansson and Nugues, 2006). Table 4 shows the performance of both parsers on the Basque test set. We see that what is gained by using a global method such as OPA is lost by sacrificing the excellent classification performance of the SVM. Possibly, better performance could be achieved by using a large-margin batch method such as *SVM*^{struct}.

Table 2: Feature sets.

	ar	**ca**	**cs**	**el**	**en**	**eu**	**hu**	**it**	**tr**	**zh**
Fine POS top	●	●	●	●	●	●	●	●	●	●
Fine POS top-1	●	●	●	●		●	●	●		
Fine POS list	●	●	●	●	●	●	●	●	●	●
Fine POS list-1	●	●	●	●	●	●	●	●	●	●
Fine POS list+1	●	●	●	●	●	●	●	●	●	●
Fine POS list+2	●	●	●	●	●	●	●	●	●	●
Fine POS list+3		●	●			●	●	●	●	
POS top	●	●	●	●	●	●	●	●	●	●
POS top-1								●		
POS list	●	●	●	●	●	●	●	●	●	●
POS list-1		●		●	●	●			●	●
POS list+1	●	●	●	●	●	●	●	●	●	●
POS list+2		●	●	●		●	●	●	●	●
POS list+3	●	●		●	●	●	●	●	●	
Features top	●	●	●	●		●	●	●	●	
Features list	●	●	●	●		●	●	●	●	
Features list-1			●	●		●		●	●	
Features list+1	●	●	●	●		●		●	●	
Features list+2	●	●		●			●	●		
Word top	●	●	●	●	●		●	●	●	●
Word top-1		●			●					
Word list	●	●	●	●	●	●	●	●	●	●
Word list-1		●	●	●	●				●	●
Word list+1	●	●				●		●		
Lemma top	●	●	●	●		●		●		
Lemma list	●	●				●		●	●	
Lemma list-1						●	●			
Relation top	●	●								
Relation top left	●	●		●				●	●	
Relation top right	●	●	●	●				●		
Relation list right						●				
Word top left	●									
Word top right	●									
Word list left						●				
POS top left				●				●		
POS top right	●			●	●					
POS list left	●	●		●	●	●	●	●	●	
Features top right						●				
Features first left						●		●		

Table 3: Summary of results.

Languages	**Unlabeled**	**Labeled**
Arabic	80.91	71.76
Basque	80.41	75.08
Catalan	88.34	83.33
Chinese	81.30	76.30
Czech	77.39	70.98
English	81.43	80.29
Greek	79.58	72.77
Hungarian	75.53	71.31
Italian	81.55	77.55
Turkish	84.80	78.46
Average result	81.12	75.78

Table 4: Accuracy by learning method.

Learning Method	**Accuracy**
OPA	75.08
SVM	75.53

4.2 Beam Width

To investigate the influence of the beam width on the performance, we measured the accuracy of a left-to-right parser on a development set for Basque (15% of the training data) as a function of the width. Table 5 shows the result. We see clearly that widening the beam considerably improves the figures, especially in the lower ranges.

Table 5: Accuracy by beam width.

Width	Accuracy
2	72.01
4	74.18
6	75.05
8	75.30
12	75.49

4.3 Direction

We also investigated the contribution of the bidirectional parsing. Table 6 shows the result of this experiment on the Basque development set (the same 15% as in 4.2). The beam width was 2 in this experiment.

Table 6: Accuracy by parsing direction.

Direction	Accuracy
Left to right	72.01
Right to left	71.02
Bidirectional	74.48

Time did not allow a full-scale experiment, but for all languages except Catalan and Hungarian, the bidirectional parsing method outperformed the unidirectional methods when trained on a 20,000-word subset. However, the gain of using bidirectional parsing may be more obvious when the treebank is small. For all languages except Czech, left-to-right outperformed right-to-left parsing.

5 Discussion

The paper describes an incremental parser that we trained to minimize the cost over sentences, rather than over parsing actions as is usually done. It was trained using the Online Passive-Aggressive method, a cost-sensitive online margin-based learning method, and shows reasonable performance and received above-average scores for most languages.

The performance of the parser (relative the other teams) was best for Basque and Turkish, which were two of the smallest treebanks. Since we found that the optimal number of iterations was 5 for Basque (the smallest treebank), we used this number for all languages since we did not have time to investigate this parameter for the other languages. This may have had a detrimental effect for some languages. We think that some of the figures might be squeezed slightly higher by optimizing learning parameters and feature sets.

This work shows that it was possible to combine approaches used by Nivre's and McDonald's parsers in a single system. While the parser is outperformed by a system based on local classifiers, we still hope that the parsing and training combination described here opens new ways in parser design and eventually leads to the improvement of parsing performance.

Acknowledgements

This work was made possible because of the annotated corpora that were kindly provided to us: (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003)

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL-X*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Schwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 2006(7):551–585.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of ICCL*.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2006. Investigating multilingual dependency parsing. In *CoNLL-X*.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency parsing with a two-stage discriminative parser. In *CoNLL-X*.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL-05*.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *CoNLL-X*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proceedings of the HLT-NAACL*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484.

Online learning for Deterministic Dependency Parsing

Prashanth Reddy Mannem
Language Technologies Research Center
IIIT-Hyderabad, India
prashanth@research.iiit.ac.in

Abstract

Deterministic parsing has emerged as an effective alternative for complex parsing algorithms which search the entire search space to get the best probable parse tree. In this paper, we present an online large margin based training framework for deterministic parsing using Nivre's Shift-Reduce parsing algorithm. Online training facilitates the use of high dimensional features without creating memory bottlenecks unlike the popular SVMs. We participated in the CoNLL Shared Task-2007 and evaluated our system for ten languages. We got an average multilingual labeled attachment score of 74.54 % (with 65.50% being the average and 80.32% the highest) and an average multilingual unlabeled attachment score of 80.30% (with 71.13% being the average and 86.55% the highest).

1 Introduction

CoNLL-X had a shared task on multilingual dependency parsing (Buchholz et al., 2006) by providing treebanks for 13 languages in the same dependency format. A look at the performance sheet in the contest shows that two systems with quite different approaches (one using deterministic parsing with SVM and the other using MIRA with nondeterministic and dynamic programming based MST approach) performed with good results (McDonald et al., 2006; Nivre et al., 2006).

More recently, deterministic parsing has generated a lot of interest because of their simplicity (Nivre, 2003). One of the main advantages of deterministic parsing lies in the ability to use the subtree information in the features to decide the next step. Parsing algorithms which search the entire space (Eisner, 1996; McDonald, 2006) are restricted in the features they use to score a relation. They rely only on the context information and not the history information to score a relation. Using history information makes the search intractable. Whereas, since deterministic parsers are at worst $O(n^2)$ (Yamada and Matsumoto, 2003) (Nivre (2003) is only $O(2n)$ in the worst case), they can use the crucial history information to make parsing decisions. So, in our work Nivre's parsing algorithm has been used to arrive at the dependency parse tree.

Popular learning algorithms for deterministic parsing like Support Vector Machines (SVM) run into memory issues for large data since they are batch learning algorithms. Though more information is available in deterministic parsing in terms of subtree information, high dimensional features can't be used due to the large training times for SVMs. This is where online methods come into the picture.

Unlike batch algorithms, online algorithms consider only one training instance at a time when optimizing parameters. This restriction to single-instance optimization might be seen as a weakness, since the algorithm uses less information about the objective function and constraints than batch algorithms. However, McDonald (2006) argues that this potential weakness is balanced by the simplicity of online learning, which allows for more streamlined training methods. This work focuses purely on online learning for deterministic parsing.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1139–1143,
Prague, June 2007. ©2007 Association for Computational Linguistics

In the remaining part of the paper, we introduce Nivre's parsing algorithm, propose a framework for online learning for deterministic parsing and present the results for all the languages with various feature models.

2 Parsing Algorithm

We used Nivre's top-down/bottom-up linear time parsing algorithm proposed in Nivre (2003). A parser configuration is represented by triples (S, I, E) where *S* is the stack (represented as a list), *I* is the list of (remaining) tokens and *E* is the set of edges for the dependency graph *D*. S is a list of partially processed tokens, whose subtrees are incomplete i.e tokens whose parent or children have not yet been established. top is the top of the stack S, $next$ is the next token in the list I.

Nivre's algorithm consists of four elementary actions $Shift$, $Left$, $Right$ and $Reduce$ to build the dependency tree from the initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$, where *W* is the input sentence. $Shift$ pushes $next$ onto the stack S. $Reduce$ pops the stack. $Right$ adds an arc from top to $next$ and pushes $next$ onto the stack S. $Left$ adds an arc from $next$ to top and pops the stack. The parser terminates when it reaches a configuration $(S, \mathbf{nil}, \mathbf{E})$ (for any list *S* and set of edges *E*).

The labels for each relation are determined after a new arc is formed (by $left$ and $right$ actions). The parser always constructs a dependency graph that is acyclic and projective. For non-projective parsing, we followed the pseudo projective parsing approach proposed by Nivre and Nilson (2005). In this approach, the training data is projectivized by a minimal transformation, lifting non-projective arcs one step at a time, and extending the arc label of the lifted arcs using the encoding scheme called HEAD+PATH. The non-projective arcs can be recovered by applying an inverse transformation to the output of the parser, using a left-to-right, top-down, breadth-first search, guided by the extended arc labels. This method has been used for all the languages.

3 Online Learning

McDonald (2005) applied online learning by scoring edges in a connected graph and finding the Maximum Spanning Tree (MST) of the graph. McDonald et al. (2005) used *Edge Based Factorization* , where the score of a dependency tree is factored as the sum of scores of all edges in the tree. Let, $\mathbf{x} = x_1 \cdots x_n$ represents a generic input sentence , and **y** represents a generic dependency tree for sentence **x**. $(i, j) \in \mathbf{y}$ denotes the presence of a dependency relation in **y** from word x_i (parent) to word x_j (child).

In Nivre's parsing algorithm the dependency graph can be viewed as a graph resulting from a set of parsing decisions (in this case *Shift*, *Reduce* , *Left & Right*) made, starting with the initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$. We define this sequence of parsing decisions as $\mathbf{d} = d_1 \cdots d_m$. So, **d** is the sequence of parsing decisions made by the parser to obtain a dependency tree **y**, from an input sentence **x**. Lets also define $\mathbf{c} = c_1 \cdots c_m$ to be the configuration sequence starting from initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$ to the final configuration $(S, \mathbf{nil}, \mathbf{E})$.

We define the score of a parsing decision for a particular configuration to be the dot product between a high dimensional feature vector (based on both the decision and the configuration) and a weight vector. So,

$$s(d_i, c_i) = \mathbf{w}\ .\ \mathbf{f}(d_i, c_i)$$

where c_i is the configuration at the i^{th} instance and d_i is any one of the four actions {Shift, Reduce, Left, Right} .

The Margin Infused Relaxed Algorithm (MIRA) proposed by Crammer et al. (2003) attempts to keep the norm of the change to the parameter vector as small as possible, subject to correctly classifying the instance under consideration with a margin at least as large as the loss of the incorrect classifications. McDonald et al. (2005) defines the loss of a dependency tree inferred by finding the Maximum Spanning Tree(MST), as the number of words that have incorrect parent (i.e the no. of edges that have gone wrong). This satisfies the global constraint that the correct set of edges will have the highest weight. However, in Nivre's algorithm, as there is no one to one correspondence between parsing decisions and the graph edges, the number of errors in the edges can't be used as a loss function as it won't reflect the exact loss in the parsing decisions. In this method of calculating the loss function based on edges, we first get the series of decisions through inference on the training data, then concat their feature vectors and finally run the normal updates with the edge based loss (since the resulting decisions will produce a parse tree). This method gave very poor results.

So we do a factored MIRA for Nivre's algorithm by factoring the output by decisions to obtain the following constraints:

$$\begin{aligned}
&min\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\| \\
&s.ts(d_i, c_i) - s(d^{'}_i, c_i) \geq 1 \\
&\forall \mathbf{c}_i \in dt(\mathbf{c}) \; and \\
&(d_i, d^{'}_i) \in (Shift, Left, Right, Reduce)
\end{aligned}$$

where d_i represents the correct decision and $d^{'}_i$ represents all the other decisions for the same configuration c_i. This states that the weight of the correct decision for a particular configuration and the weight of all other decisions must be separated by a margin of 1. For every sentence in the training data, starting with the initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$, weights are adjusted to satisfy the above constraints before proceeding to the next correct configuration. This process is repeated till we reach the final configuration $(\mathbf{S}, \mathbf{nil}, \mathbf{E})$.

4 Features

The two central elements in any configuration are the token on the top of the stack (*t*) and the next input token (*n*),the tokens which may be connected by a dependency relation in the current configuration. We categorize our features into $basic$, $context$, $history$ and $in - between$ feature sets. The *basic* feature set contains information about these two tokens t and n. This includes unigram, bigram combinations of the word forms (FORM), root word (LEMMA), features (FEATS) and the part-of-speech tags (both CPOS and POS) of these words. The coarse POS tag (CPOS) is useful and helps solve data sparseness to some extent.

The existence or non-existence of a relation between two words is heavily dependent on the words surrounding t and n which is the contextual information. The *context* feature set has the information about the surrounding words $t-1$, $t+1$, $n-1$, $n+1$, $n+2$, $n+3$. Unigram and trigram combinations (with t and n) of the lexical items, POS tags, CPOS tags of these words are part of this context feature set. We also included the second topmost element in the stack ($st-1$) word too.

The third feature set, which is the *history* feature set contains the info about the subtree at a particular parser state. One of the advantages of using deterministic parsing algorithm over nondeterministic algorithm is that history can be used as features. History features have information about the *Parent* (par), *Left Sibling* (ls) and *Right Sibling* (rs) of t. Unigram and trigram combinations (with t and n) of POS, CPOS, DEPREL tags of these words are included in the $History$ Features.

The features in the $in - between$ feature set take the form of POS and CPOS trigrams: the POS/CPOS of t, that of the word in between and that of n.

All the features in these feature sets are conjoined with distance between t & n and the parsing decision. We experimented with a combination of these feature sets in our training. We define feature models ϕ_1, ϕ_2 and ϕ_3 for our experiments. ϕ_1 is a combination on $basic$ and $context$ feature sets. ϕ_2 is a mixture of $basic$, $context$ and $in - between$ feature sets whereas ϕ_3 contains $basic$, $context$ and $history$ feature sets. The feature models ϕ_{1-3} are the same for all the languages.

5 Results and Discussion

The system with online learning and Nivre's parsing algorithm was trained on the data released by CoNLL Shared Task Organizers for all the ten languages (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003). We evaluated our system using the standard evaluation script provided by the organizers (Nivre et al., 2007). The evaluation metrics are Unlabeled Attachment Score(UAS) and Labeled Attachment Score(LAS).

The results of our system with various feature models are listed in Table 1[1]. The history information in ϕ_3 contributed to a marginal improvement in accuracy of Hungarian, Italian and Turkish. Whereas, Arabic, Catalan, Czech, English, Greek

[1] Results aren't available for the models with a '-' mark.

Language	ϕ_1		ϕ_2		ϕ_3	
	LAS	UAS	LAS	UAS	LAS	UAS
$Arabic$	71.55	81.56	**72.05**	**81.93**	71.66	81.30
$Basque$	**66.35**	**73.71**	65.64	72.86	64.56	71.69
$Catalan$	84.45	89.65	**84.47**	**89.81**	-	-
$Chinese$	**74.06**	**79.09**	73.76	78.84	72.93	77.52
$Czech$	70.49	77.05	**70.68**	**77.20**	-	-
$English$	81.19	82.41	**81.55**	**82.81**	-	-
$Greek$	71.52	78.77	**71.69**	**78.89**	71.46	78.48
$Hungarian$	70.42	75.01	70.94	75.39	**71.05**	**75.54**
$Italian$	78.30	82.54	78.67	82.91	**79.18**	**83.38**
$Turkish$	76.42	82.74	76.48	82.85	**77.29**	**83.58**

Table 1: Results of Online learning with Nivre's parsing algorithm for feature models ϕ_1, ϕ_2, ϕ_3

got their highest accuracies with feature model ϕ_2 containing $basic$, $context$ and $in-between$ feature sets. The rest of the languages, Basque and Chinese achieved highest accuracies with ϕ_1. But, a careful look at the results table shows that there isn't any significant difference in the accuracies of the system across different feature models. This is true for all the languages. The feature models ϕ_2 and ϕ_3 did not show any significant difference in accuracies even though they contain more information. Feature model ϕ_1 with $basic$ and $context$ feature sets has achieved good accuracies.

5.1 K-Best Deterministic Parsing

The deterministic parsing algorithm does not handle ambiguity. By choosing a single parser action at each opportunity, the input string is parsed deterministically and a single dependency tree is built during the parsing process from beginning to end (no other trees are even considered). A simple extension to this idea is to eliminate determinism by allowing the parser to choose several actions at each opportunity, creating different action sequences that lead to different parse trees. Since a score is assigned to every parser action, the score of a parse tree can be computed simply as the average of the scores of all actions that resulted in that parse tree (the derivation of the tree). We performed a beam search by carrying out a K-best search through the set of possible sequences of actions as proposed by Johansson and Nugues (2006). However, this did not increase the accuracy. Moreover, with larger values of K, there was a decrease in the parsing accuracy. The best-first search proposed by Sagae and Lavie (2006) was also tried out but there was similar drop in accuracy.

6 Conclusion

The evaluation shows that the labeled pseudo projective deterministic parsing with online learning gives competitive parsing accuracy for most of the languages involved in the shared task. The level of accuracy varies considerably between the languages. Analyzing the results and the effects of various features with online learning will be an important research goal in the future.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL).*

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2003. Online passive aggressive algorithms. In *Proceedings of Neural Information Processing Systems (NIPS)*.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2006. Investigating multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 206–210.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

R. McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

J. Nivre, J. Hall, J. Nilson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of International Workshop on Parsing Technologies*, pages 149–160.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

K. Sagae and A. Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT-2003*.

Covington Variations

Svetoslav Marinov
School of Humanities and Informatics,
University College Skövde, 54128 Skövde &
GSLT, Göteborg University, 40530 Göteborg,
Sweden
Svetoslav.Marinov@his.se

Abstract

Three versions of the Covington algorithm for non-projective dependency parsing have been tested on the ten different languages for the Multilingual track of the CoNLL-X Shared Task. The results were achieved by using only information about heads and daughters as features to guide the parser which obeys strict incrementality.

1 Introduction

In this paper we focus on two things. First, we investigate the impact of using different flavours of Covington's algorithm (Covington, 2001) for non-projective dependency parsing on the ten different languages provided for CoNLL-X Shared Task (Nivre et al., 2007). Second, we test the performance of a pure grammar-based feature model in strictly incremental fashion. The grammar model relies only on the knowledge of heads and daughters of two given words, as well as the words themselves, in order to decide whether they can be linked with a certain dependency relation. In addition, none of the three parsing algorithms guarantees that the output dependency graph will be projective.

2 Covington's algorithm(s)

In his (2001) paper, Covington presents a "fundamental" algorithm for dependency parsing, which he claims has been known since the 1960s but has, up to his paper-publication, not been presented systematically in the literature. We take three of its flavours, which enforce uniqueness (a.k.a. single-headedness) but do not observe projectivity. The algorithms work one word at a time and attempt to build a connected dependency graph with only a single left-to-right pass through the input. The three flavours are: Exhaustive Search, Head First with Uniqueness (ESHU), Exhaustive Search Dependents First with Uniqueness (ESDU) and List-based search with Uniqueness (LSU).

```
ESHU
for $i$ = 1 to $n$
    for $j$ = $i$-1 downto 0
        if HEAD?($j$,$i$)
            LINK($j$,$i$)
        if HEAD?($i$,$j$)
            LINK($i$,$j$)
```

```
ESDU
for $i$ = 1 to $n$
    for $j$ = $i$-1 downto 0
        if HEAD?($i$,$j$)
            LINK($i$,$j$)
        if HEAD?($j$,$i$)
            LINK($j$,$i$)
```

The yes/no function **HEAD?**(*w1*,*w2*), checks whether a word *w1* can be a head of a word *w2* according to a grammar *G*. It also respects the single-head and no-cycle conditions. The **LINK**(*w1*,*w2*) procedure links word *w1* as the head of word *w2* with a dependency relation as proposed by *G*. When traversing *Headlist* and *Wordlist* we start with the last word added. (Nivre, 2007) describes an optimized version of Covington's algorithm implemented in MaltParser (Nivre, 2006) with a running time $c(\frac{n^2}{2} - \frac{n}{2})$ for an n-word sentence, where c is some constant time in which the **LINK** operation can be performed. However, due to time constraints, we will not bring this version of the algorithm into focus, but see some preliminary remarks on it with respect to our parsing model in 6.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1144–1148,
Prague, June 2007. ©2007 Association for Computational Linguistics

```
LSU[1]
Headlist := []
Wordlist := []
while (!end-of-sentence)
   W := next input word;
   foreach D in Headlist
      if HEAD?(W,D)
         LINK(W,D);
         delete D from Headlist;
   end
   foreach H in Wordlist
      if HEAD?(H,W)
         LINK(H,W);
         terminate this foreach loop;
   end
   if no head for W was found then
      Headlist := W + Headlist;
   end
   Wordlist := W + Wordlist;
end
```

3 Classifier as an Instant Grammar

The **HEAD?** function in the algorithms presented in 2, requires an "instant grammar" (Covington, 2001) of some kind, which can tell the parser whether the two words under scrutiny can be linked and with what dependency relation. To satisfy this requirement, we use TiMBL - a Memory-based learner (Daelemans et al., 2004) - as a classifier to predict the relation (if any) holding between the two words.

Building heavily on the ideas of History-based parsing (Black et al., 1993; Nivre, 2006), training the parser means essentially running the parsing algorithms in a learning mode on the data in order to gather training instances for the memory-based learner. In a learning mode, the **HEAD?** function has access to a fully parsed dependency graph. In the parsing mode, the **HEAD?** function in the algorithms issues a call to the classifier using features from the parsing history (i.e. a partially built dependency graph PG).

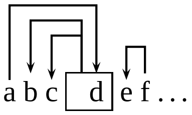
Given words i and j to be linked, and a PG, the call to the classifier is a feature vector $\Phi(i,j,PG)$ = $(\phi_1,\ldots,\phi_m)$ (cf. (Nivre, 2006; Nivre, 2007)). The classifier then attempts to map this feature vector to any of predefined classes. These are all the dependency relations, as defined by the treebank and the class "NO" in the cases where no link between the two words is possible.

4 The Grammar model

The features used in our history-based model are restricted only to the partially built graph PG. We call this model a pure grammar-based model since the only information the parsing algorithms have at their disposal is extracted from the graph, such as the head and daughters of the current word. Preceding words not included in the PG as well as words following the current word are not available to the algorithm. In this respect such a model is very restrictive and suffers from the pitfalls of the incremental processing (Nivre, 2004).

The motivation for the chosen model, was to approximate a Data Oriented Parsing (DOP) model (e.g. (Bod et al., 2003)) for Dependency Grammar. Under DOP, analyses of new sentences are produced by combining previously seen tree fragments. However, the tree fragments under the original DOP model are static, i.e. we have a corpus of all possible subtrees derived from a treebank. Under our approach, these tree fragments are built dynamically, as we try to parse the sentence. Because of the chosen DOP approximation, we have not included information about the preceding and following words of the two words to be linked in our feature model.

To exemplify our approach, (1) shows a partially build graph and all the words encountered so far and Fig. 1 shows two examples of the tree-building operations for linking words f and d, and f and a.

(1)


Given two words i and j to be linked with a dependency relation, such that word j precedes word i, the following features describe the models on which the algorithms have been trained and tested:

Word form: i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$
Lemma (if available): i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$

[1] Covington adds *W* to the *Wordlist* as soon as it has been seen, however we have chosen to wait until after all tests have been completed.

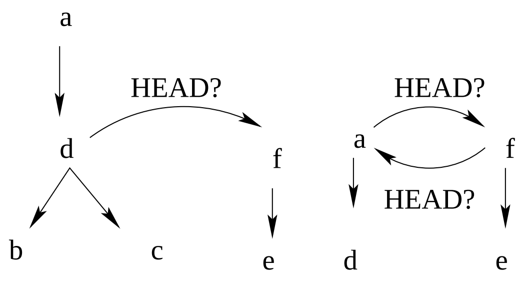


Figure 1: Application of the **HEAD?** function on an input from the PG in (1)

Part-of-Speech: $i, j, ds(i), ds(j), h(j/i), h(h(j/i))$
Dependency type: i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$
Features (if available): i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$

$ds(i)$ means any two daughters (if available) of word i, $h(i/j)$ refers to the head of word i or word j, depending on the direction of applying the **HEAD?** function (see Fig 1) and $h(h(i/j))$ stands for the head of the head of word i or word j.

The basic model, which was used for the largest training data sets of Czech and Chinese, includes only the first four features in every category. A larger model used for the datasets of Catalan and Hungarian adds the $h(j/i)$ feature from every category. The enhanced model used for Arabic, Basque, English, Greek, Italian and Turkish uses the full set of features. This tripartite division of models was motivated only by time- and resource-constraints. The simplest model is for Chinese and uses only 5 features while the enhanced model for Arabic for example uses a total of 39 features.

5 Results and Setup

Table 1 summarizes the results of testing the three algorithms on the ten different languages.

The parser was written in C#. Training and testing were performed on a MacOSX 10.4.9 with 2GHz Intel Core2Duo processor and 1GB memory, and a Dell Dimension with 2.80GHz Pentium 4 processor and 1GB memory running Mepis Linux. TiMBL was run in client-server mode with default settings (IB1 learning algorithm, extrapolation from the most similar example, i.e. $k = 1$, initiated with the command “Timbl -S <portnumber> -f <training_file>”). Additionally, we attempted to use Support Vector Machines (SVM) as an alternative classifier. However, due to the long training time, results from using SVM were not included but training an SVM classifier for some of the languages has started.

	ESHU	ESDU	LSU
Arabic	LA: 53.72 UA: 63.58	LA: **54.00** UA: 63.76	LA: 53.86 UA: **63.78**
Basque	LA: 49.52 UA: 56.83	LA: 50.20 UA: 57.81	LA: **51.24** UA: **58.53**
Catalan	LA: 69.56 UA: 74.32	LA: **69.80** UA: **74.46**	LA: 69.42 UA: 74.22
Chinese	LA: 47.57 UA: 53.46	LA: **50.61** UA: **56.75**	LA: 49.82 UA: 56.02
Czech	LA: 44.41 UA: 49.20	LA: **53.66** UA: **60.01**	LA: 53.47 UA: 59.55
English	LA: 51.05 UA: 53.41	LA: 51.35 UA: 53.65	LA: **52.11** UA: **54.33**
Greek	LA: 54.68 UA: 61.55	LA: 54.62 UA: 61.45	LA: **55.02** UA: **61.80**
Hungarian	LA: 44.34 UA: 50.12	LA: **45.11** UA: **50.78**	LA: 44.57 UA: 50.46
Italian	LA: **61.60** UA: **67.01**	LA: 60.95 UA: 66.25	LA: 61.52 UA: 66.39
Turkish	LA: 55.57 UA: 62.13	LA: **57.01** UA: **63.77**	LA: 56.59 UA: 63.17

Table 1: Test results for the 10 languages. LA is the Labelled Attachment Score and UA is the Unlabelled Attachment Score

6 Discussion

Before we attempt a discussion on the results presented in Table 1, we give a short summary of the basic word order typology of these languages according to (Greenberg, 1963). Table 2 shows whether the languages are SVO (subject-verb-object) or SOV (subject-object-verb), or VSO (verb-subject-object); contain Pr (prepositions) or Po (postpositions); NG (noun precedes genitive) or GN (genitive precedes noun); AN (adjective precedes noun) or NA (noun precedes adjective).

[2] Greenberg had give *varying* for the word-order typology of English. However, we trusted our own intuition as well as the hint of one of the reviewers.

Arabic	VSO	Pr	NG	NA
Basque	SOV	Po	GN	NA
Catalan	SVO	Pr	NG	NA
Chinese	SVO	Po	GN	AN
Czech	SVO	Pr	NG	AN
English[2]	SVO	Pr	GN	AN
Greek	SVO	Pr	NG	AN
Hungarian	SOV	Po	GN	AN
Italian	SVO	Pr	NG	NA
Turkish	SOV	Po	?	AN

Table 2: Basic word order typology of the ten languages following Greenberg's Universals

Looking at the data in Table 1, several observations can be made. One is the different performance of languages from the same language family, i.e. Italian, Greek and Catalan. However, the head-first (ESHU) algorithm presented better than the dependents-first (ESDU) one in all of these languages. The SOV languages like Hungarian, Basque and Turkish had preference for the dependent's first algorithms (ESDU and LSU). The ESDU algorithm also fared better with the SVO languages, except for Italian.

However, the Greenberg's basic word order typology cannot shed enough light into the performance of the three parsing algorithms. One question that pops up immediately is whether a different feature-model using the same parsing algorithms would achieve similar results. Can the different performance be attributed to the treebank annotation? Would another classifier fare better than the Memory-based one? These questions remain for future research though.

Finally, for the Basque data we attempted to test the optimized version of the Covington algorithm (Nivre, 2007) against the three other versions discussed here. Additionally, since our feature vectors differed from those described in (Nivre, 2007), head-dependent-features vs. j-i-features, we changed them so that all the four algorithms send a similar feature vector, j-i-features, to the classifier. The preliminary result was that Nivre's version was the fastest, with fewer calls to the **LINK** procedure and with the smallest training data-set. However, all the four algorithms showed about 20% decrease in LA/UA scores.

Our first intuition about the results from the tests done on all the 10 languages was that the classification task suffered from a highly skewed class distribution since the training instances that correspond to a dependency relation are largely outnumbered by the "NO" class (Canisius et al., 2006). The recall was low and we expected the classifier to be able to predict more of the required links. However, the results we got from additional optimizations we performed on Hungarian, following recommendation from the anonymous reviewers, may lead to a different conclusion. The chosen grammar model, relying only on connecting dynamically built partial dependency graphs, is insufficient to take us over a certain threshold.

7 Conclusion

In this paper we showed the performance of three flavours of Covington's algorithm for non-projective dependency parsing on the ten languages provided for the CoNLL-X Shared Task (Nivre et al., 2007). The experiment showed that given the grammar model we have adopted it does matter which version of the algorithm one uses. The chosen model, however, showed a poor performance and suffered from two major flaws - the use of only partially built graphs and the pure incremental processing. It remains to be seen how these parsing algorithms will perform in a parser, with a much richer feature model and whether it is worth using different flavours when parsing different languages or the differences among them are insignificant.

Acknowledgements

We would like to thank the two anonymous reviewers for their valuable comments. We are grateful to Joakim Nivre for discussion on the Covington algorithm, Bertjan Busser for help with TiMBL, Antal van den Bosch for help with paramsearch, Matthew Johnson for providing the necessary functionality to his .NET implementation of SVM and Patrycja Jabłońska for discussion on the Greenberg's Universals.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

Ezra Black, Frederick Jelinek, John D. Lafferty, David M. Magerman, Robert L. Mercer, and Salim Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *Meeting of the Association for Computational Linguistics*, pages 31–37.

R. Bod, R. Scha, and K. Sima'an, editors. 2003. *Data Oriented Parsing*. CSLI Publications, Stanford University, Stanford, CA, USA.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

Sander Canisius, Toine Bogers, Antal van den Bosch, Jeroen Geertzen, and Erik Tjong Kim Sang. 2006. Dependency Parsing by Inference over High-recall Dependency Predictions. In *CoNLL-X Shared Task on Multitlingual Dependency Parsing*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Michael A. Covington. 2001. A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102, Athens, Georgia, USA.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. Timbl: Tilburg memory based learner, version 5.1, reference guide. Technical report, ILK Technical Report 04-02, available from http://ilk.uvt.nl/downloads/pub/papers/ilk0402.pdf.

Joseph H. Greenberg. 1963. *Universals of Language*. London: MIT Press.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Joakim Nivre. 2004. Incrementality in Deterministic Dependency Parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together*, Workshop at ACL-2004, pages 50–57, Barcelona, Spain, July, 25.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Joakim Nivre. 2007. Incremental Non-Projective Dependency Parsing. In *Proceedings of NAACL-HLT 2007*, Rochester, NY, USA, April 22–27.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

A Multilingual Dependency Analysis System using Online Passive-Aggressive Learning

Le-Minh Nguyen, Akira Shimazu, and Phuong-Thai Nguyen
Japan Advanced Institute of Science and Technology (JAIST)
Asahidai 1-1, Nomi, Ishikawa, 923-1292 Japan
{nguyenml,shimazu,thai}@jaist.ac.jp

Xuan-Hieu Phan
Tohoku University
Aobayama 6-3-09, Sendai, 980-8579, Japan
hieuxuan@ecei.tohoku.ac.jp

Abstract

This paper presents an online algorithm for dependency parsing problems. We propose an adaptation of the passive and aggressive online learning algorithm to the dependency parsing domain. We evaluate the proposed algorithms on the 2007 CONLL Shared Task, and report errors analysis. Experimental results show that the system score is better than the average score among the participating systems.

1 Introduction

Research on dependency parsing is mainly based on machine learning methods, which can be called history-based (Yamada and Matsumoto, 2003; Nivre et al., 2006) and discriminative learning methods (McDonald et al., 2005a; Corston-Oliver et al., 2006). The learning methods using in discriminative parsing are Perceptron (Collins, 2002) and online large-margin learning (MIRA) (Crammer and Singer, 2003).

The difference of MIRA-based parsing in comparison with history-based methods is that the MIRA-based parser were trained to maximize the accuracy of the overall tree. The MIRA based parsing is close to maximum-margin parsing as in Taskar et al. (2004) and Tsochantaridis et al. (2005) for parsing. However, unlike maximum-margin parsing, it is not limited to parsing sentences of 15 words or less due to computation time. The performance of MIRA based parsing achieves the state-of-the-art performance in English data (McDonald et al., 2005a; McDonald et al., 2006).

In this paper, we propose a new adaptation of online larger-margin learning to the problem of dependency parsing. Unlike the MIRA parser, our method does not need an optimization procedure in each learning update, but users only an update equation. This might lead to faster training time and easier implementation.

The contributions of this paper are two-fold: First, we present a training algorithm called PA learning for dependency parsing, which is as easy to implement as Perceptron, yet competitive with large margin methods. This algorithm has implications for anyone interested in implementing discriminative training methods for any application. Second, we evaluate the proposed algorithm on the multilingual data task as well as the domain adaptation task (Nivre et al., 2007).

The remaining parts of the paper are organized as follows: Section 2 proposes our dependency parsing with Passive-Aggressive learning. Section 3 discusses some experimental results and Section 4 gives conclusions and plans for future work.

2 Dependency Parsing with Passive-Aggressive Learning

This section presents the modification of Passive-Aggressive Learning (PA) (Crammer et al., 2006) for dependency parsing. We modify the PA algorithm to deal with structured prediction, in which our problem is to learn a discriminant function that maps an input sentence x to a dependency tree y. Figure 1 shows an example of dependency parsing which depicts the relation of each word to another word within a sentence. There are some algorithms

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1149–1155,
Prague, June 2007.

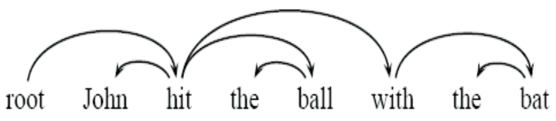


Figure 1: This is an example of dependency tree

to determine these relations of each word to another words, for instance, the modified CKY algorithm (Eisner, 1996) is used to define these relations for a given sentence.

2.1 Parsing Algorithm

Dependency-tree parsing as the search for the maximum spanning tree (MST) in a graph was proposed by McDonald et al. (2005b). In this subsection, we briefly describe the parsing algorithms based on the first-order MST parsing. Due to the limitation of participation time, we only applied the first-order decoding parsing algorithm in CONLL-2007. However, our algorithm can be used for the second order parsing.

Let the generic sentence be denoted by x ; the ith word of x is denoted by w_i. The generic dependency tree is denoted by y. If y is a dependency tree for sentence x, we write $(i, j) \in y$ to indicate that there is a directed edge from word xw_i to word xw_j in the tree, that is, xw_i is the parent of xw_j . $T = \{(x_t, y_t)\}_{t=1}^{n}$ denotes the training data. We follow the edge based factorization method of Eisner (Eisner, 1996) and define the score of a dependency tree as the sum of the score of all edges in the tree,

$$s(x,y) = \sum_{(i,j)\in y} s(i,j) = \sum_{(i,j)\in y} \mathbf{w} \cdot \Phi(i,j) \quad (1)$$

where $\Phi(i, j)$ is a high-dimensional binary feature representation of the edge from xw_i to xw_j . For example in Figure 1, we can present an example $\Phi(i, j)$ as follows;

$$\Phi(i,j) = \begin{cases} 1 \,\mathrm{if}\, xw_i =' hit' \, and \, xw_j =' ball' \\ 0 \,\mathrm{otherwise} \end{cases}$$

The basic question must be answered for models of this form: how to find the dependency tree y with the highest score for sentence x? The two algorithms we employed in our dependency parsing model are the Eisner parsing (Eisner, 1996) and Chu-Liu's algorithm (Chu and Liu, 1965). The algorithms are commonly used in other online-learning dependency parsing, such as in (McDonald et al., 2005a).

In the next subsection we will address the problem of how to estimate the weight w_i associated with a feature Φ_i in the training data using an online PA learning algorithm.

2.2 Online PA Learning

This section presents a modification of PA algorithm for structured prediction, and its use in dependency parsing. The Perceptron style for natural language processing problems as initially proposed by (Collins, 2002) can provide state of the art results on various domains including text chunking, syntactic parsing, etc. The main drawback of the Perceptron style algorithm is that it does not have a mechanism for attaining the maximize margin of the training data. It may be difficult to obtain high accuracy in dealing with hard learning data. The structured support vector machine (Tsochantaridis et al., 2005) and the maximize margin model (Taskar et al., 2004) can gain a maximize margin value for given training data by solving an optimization problem (i.e quadratic programming). It is obvious that using such an optimization algorithm requires much computational time. For dependency parsing domain, McDonald et al (2005a) modified the MIRA learning algorithm (McDonald et al., 2005a) for structured domains in which the optimization problem can be solved by using Hidreth's algorithm (Censor and Zenios, 1997), which is faster than the quadratic programming technique. In contrast to the previous method, this paper presents an online algorithm for dependency parsing in which we can attain the maximize margin of the training data without using optimization techniques. It is thus much faster and easier to implement. The details of PA algorithm for dependency parsing are presented below.

Assume that we are given a set of sentences x_i and their dependency trees y_i where $i = 1, ..., n$. Let the feature mapping between a sentence x and a tree y be: $\Phi(x, y) = \Phi_1(x, y), \Phi_2(x, y), ..., \Phi_d(x, y)$ where each feature mapping Φ_j maps (x, y) to a real value. We assume that each feature $\Phi(x, y)$ is associated with a weight value. The goal of PA learning for dependency parsing is to obtain a parameter w that minimizes the hinge-loss function and the margin of learning data.

```
1  Input:$S = \{(x_i; y_i), i = 1, 2, ..., n\}$ in which
   $x_i$ is the sentence and $y_i$ is a dependency tree
2  Aggressive parameter $C$
3  Output: the PA learning model
4  Initialize: $w_1 = (0, 0, ..., 0)$
5  for t=1, 2... do
6      Receive an sentence $x_t$
7      Predict $y_t^* = \arg\max_{y \in Y}(\mathbf{w_t} \cdot \Phi(x_t, y_t))$
8      Suffer loss: $l_t =$
       $\mathbf{w_t} \cdot \Phi(x_t, y_t^*) - \mathbf{w_t} \cdot \Phi(x_t, y_t) + \sqrt{\rho(y_t, y_t^*)}$
9      Set:

         PA: $\tau_t = \frac{l_t}{||\Phi(x_t, y_t^*) - \Phi(x_t, y_t)||^2}$
         PA-I: $\tau_t = \min\{C, \frac{l_t}{||\Phi(x_t, y_t^*) - \Phi(x_t, y_t)||^2}\}$
         PA-II: $\tau_t = \frac{l_t}{||\Phi(x_t, y_t^*) - \Phi(x_t, y_t)||^2 + \frac{1}{2C}}$

       Update:
       $w_{t+1} = w_t + \tau_t(\Phi(x_t, y_t) - \Phi(x_t, y_t^*))$
10 end
```

Algorithm 1: The Passive-Aggressive algorithm for dependency parsing.

Algorithm 1 shows the PA learning algorithm for dependency parsing in which its three variants are different only in the update formulas. In Algorithm 1, we employ two kinds of argmax algorithms: The first is the decoding algorithm for projective language data and the second one is for non-projective language data. Algorithm 1 shows (line 8) $p(y, y_t)$ is a real-valued loss for the tree y_t relative to the correct tree y. We define the loss of a dependency tree as the number of words which have an incorrect parent. Thus, the largest loss a dependency tree can have is the length of the sentence. The similar loss function is designed for the dependency tree with labeled. Algorithm 1 returns an averaged weight vector: an auxiliary weight vector v is maintained that accumulates the values of w after each iteration, and the returned weight vector is the average of all the weight vectors throughout training. Averaging has been shown to help reduce overfitting (McDonald et al., 2005a; Collins, 2002). It is easy to see that the main difference between the PA algorithms and the Perceptron algorithm (PC) (Collins, 2002) as well as the MIRA algorithm (McDonald et al., 2005a) is in line 9. As we can see in the PC algorithm, we do not need the value τ_t and in the MIRA algorithm we need an optimization algorithm to compute τ_t. We also have three updated formulations for obtaining τ_t in Line 9. In the scope of this paper, we only focus on using the second update formulation (PA-I method) for training dependency parsing data.

2.3 Feature Set

We denote p-word: word of parent node in dependency tree. c-word: word of child node. p-pos: POS of parent node. c-pos: POS of child node. p-pos+1: POS to the right of parent in sentence. p-pos-1: POS to the left of parent. c-pos+1: POS to the right of child. c-pos-1: POS to the left of child. b-pos: POS of a word in between parent and child nodes. The features used in our system are described below.

p-word,p-pos
p-word
p-pos
c-word, c-pos
c-word
c-pos

Table 1: Feature Set 1: Basic Unit-gram features

p-word, p-pos, c-word, c-pos
p-pos, c-word, c-pos
p-word, c-word, c-pos
p-word, p-pos, c-pos
p-word, p-pos, c-word
p-word, c-word
p-pos, c-pos

Table 2: Feature Set 2: Basic bi-gram features

p-pos, b-pos, c-pos
p-pos, p-pos+1, c-pos-1, c-pos
p-pos-1, p-pos, c-pos-1, c-pos
p-pos, p-pos+1, c-pos, c-pos+1
p-pos-1, p-pos, c-pos, c-pos+1

Table 3: Feature Set 3: In Between POS Features and Surrounding Word POS Features

- Tables 1 and 2 show our basic features. These features are added for entire words as well as for the 5-gram prefix if the word is longer than 5 characters.

- In addition to these features shown in Table 1, the morphological information for each pair of words p-word and c-word are represented. In addition, we also add the conjunction morphological information of p-word and c-word. We do not use the LEMMA and CPOSTAG information in our set features. The morphological information are obtained from FEAT information.

- Table 3 shows our Feature set 3 which take the form of a POS trigram: the POS of the parent, of the child, and of a word in between, for all words linearly between the parent and the child. This feature was particularly helpful for nouns identifying their parent (McDonald et al., 2005a).

- Table 3 also depicts these features taken the form of a POS 4-gram: The POS of the parent, child, word before/after parent and word before/after child. The system also used back-off features with various trigrams where one of the local context POS tags was removed.

- All features are also conjoined with the direction of attachment, as well as the distance between the two words being attached.

3 Experimental Results and Discussion

We test our parsing models on the CONLL-2007 (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003) data set on various languages including Arabic, Basque, Catalan, Chinese, English, Italian, Hungarian, and Turkish. Each word is attached by POS tags for each sentence in both the training and the testing data. Table 4 shows the number of training and testing sentences for these languages. The table shows that the sentence length in Arabic data is largest and its size of training data is smallest. These factors might be affected to the accuracy of our proposed algorithm as we will discuss later.

The training and testing were conducted on a pentium IV at 4.3 GHz. The detailed information about the data are shown in the CONLL-2007 shared task. We applied non-projective and projective parsing along with PA learning for the data in CONLL-2007.

Table 5 reports experimental results by using the first order decoding method in which an MST parsing algorithm (McDonald et al., 2005b) is applied for non-projective parsing and the Eisner's method is used for projective language data. In fact, in our method we applied non-projective parsing for the Italian data, the Turkish data, and the Greek data. This was because we did not have enough time to train all training data using both projective and non-projective parsing. This is the problem of discriminative learning methods when performing on a large set of training data. In addition, to save time in training we set the number of best trees k to 1 and the parameter C is set to 0.05.

Table 5 shows the comparison of the proposed method with the average, and three top systems on the CONLL-2007. As a result, our method yields results above the average score on the CONLL-2007 shared task (Nivre et al., 2007).

Table 5 also indicates that the Basque results obtained a lower score than other data. We obtained 69.11 UA score and 58.16 LA score, respectively. These are far from the results of the Top3 scores (81.13 and 75.49). We checked the outputs of the Basque data to understand the main reason for the errors. We see that the errors in our methods are usually mismatched with the gold data at the labels "ncmod" and "ncsubj". The main reason might be that the application of projective parsing for this data in both training and testing is not suitable. This was because the number of sentences with at least 1 non projective relation in the data is large (26.1).

The Arabic score is lower than the scores of other data because of some difficulties in our method as follows. Morphological and sentence length problems are the main factors which affect the accuracy of parsing Arabic data. In addition, the training size in the Arabic is also a problem for obtaining a good result. Furthermore, since our tasks was focused on improving the accuracy of English data, it might be unsuitable for other languages. This is an imbalance

Languages	Training size	Tokens size	tokens-per-sent	% of NPR	% of-sentence AL-1-NPR
Arabic	2,900	112,000	38.3	0.4	10.1
Basque	3,200	51,000	15.8	2.9	26.2
Catalan	15,000	431,000	28.8	0.1	2.9
Chinese	57,000	337,000	5.9	0.0	0.0
Czech	25,400	432,000	17.0	1.9	23.2
English	18,600	447,000	24.0	0.3	6.7
Greek	2,700	65,000	24.2	1.1	20.3
Hungarian	6,000	132,000	21.8	2.9	26.4
Italian	3,100	71,000	22.9	0.5	7.4
Turkish	5,600	65,000	11.6	0.5	33.3

Table 4: The data used in the multilingual track (Nivre et al., 2007). NPR means non-projective-relations. AL-1-NPR means at-least-least 1 non-projective relation.

problem in our method. Table 5 also shows the comparison of our system to the average score and the Top3 scores. It depicts that our system is accurate in English data, while it has low accuracy in Basque and Arabic data.

We also evaluate our models in the domain adaptation tasks. This task is to adapt our model trained on PennBank data to the test data in the Biomedical domain. The pchemtb-closed shared task (Marcus et al., 1993; Johansson and Nugues, 2007; Kulick et al., 2004) is used to illustrate our models. We do not use any additional unlabeled data in the Biomedical domain. Only the training data in the PennBank is used to train our model. Afterward, we selected carefully a suitable parameter using the development test set. We set the parameter C to 0.01 and select the non projective parsing for testing to obtain the highest result in the development data after performing several experiments. After that, the trained model was used to test the data in Biomedical domain. The score (UA=82.04; LA=79.50) shows that our method yields results above the average score (UA=76.42; LA=73.03). In addition, it is officially coming in 4th place out of 12 teams and within 1.5% of the top systems.

The good result of performing our model in another domain suggested that the PA learning seems sensitive to noise. We hope that this problem is solved in future work.

4 Conclusions

This paper presents an online algorithm for dependency parsing problem which have tested on various language data in CONLL-2007 shared task. The performance in English data is close to the Top3 score. We also perform our algorithm on the domain adaptation task, in which we only focus on the training of the source data and select a suitable parameter using the development set. The result is very good as it is close to the Top3 score of participating systems. Future work will also be focused on extending our method to a version of using semi-supervised learning that can efficiently be learnt by using labeled and unlabeled data. We hope that the application of the PA algorithm to other NLP problems such as semantic parsing will be explored in future work.

Acknowledgments

We would like to thank D. Yuret for his helps in checking errors of my parser's outputs. We would like to thank Vinh-Van Nguyen his helps during the revision process and Mary Ann Mooradian for correcting the paper.

We would like to thank to anonymous reviewers for helpful discussions and comments on the manuscript. Thank also to Sebastian Riedel for checking the issues raised in the reviews.

The work on this paper was supported by a Monbukagakusho 21st COE Program.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

Languages	Unlabled Accuracy					Labeled Accuracy					NTeams
	PA-I	Average	Top3	Top2	Top1	PA-I	Average	Top3	Top2	Top1	
Arabic	73.46	78.84	84.21	85.81	86.09	68.34	74.75	83.0	75.08	76.52	20
Basque	69.11	75.15	81.13	81.93	81.13	58.16	68.06	75.49	75.73	76.92	20
Catalan	88.12	87.98	93.12	93.34	93.40	83.23	79.85	87.90	88.16	88.70	20
Chinese	84.05	81.98	87.91	88.88	88.94	79.77	76.59	83.51	83.84	84.69	21
Czech	80.91	77.56	84.19	85.16	86.28	72.54	70.12	77.98	78.60	80.19	20
English	88.01	82.67	89.87	90.13	90.63	86.73	80.95	88.41	89.01	89.61	23
Greek	77.56	77.78	81.37	82.04	84.08	70.42	70.22	74.42	74.65	76.31	20
Hungarian	78.13	76.34	82.49	83.51	83.55	68.12	71.49	78.09	79.53	80.27	21
Italian	80.40	82.45	87.68	87.77	87.91	75.06	78.06	78.09	79.53	80.27	20
Turkish	80.19	73.19	85.77	85.77	86.22	67.63	73.19	79.24	79.79	79.81	20
Multilingual-average	79.99	71.13	85.62	85.71	86.55	72.52	65.77	79.90	80.28	80.32	23
pchemtb-closed	82.04	76.42	83.08	83.38	83.42	79.50	73.03	80.22	80.40	81.06	8

Table 5: Dependency accuracy in the CONLL-2007 shared task.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

Y. Censor and S.A. Zenios. 1997. Parallel optimization: theory, algorithms, and applications. In *Oxford University Press*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. In *Science Sinica*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

S. Corston-Oliver, A. Aue, K. Duh, , and E. Ringger. 2006. Multilingual dependency parsing using bayes point machines. In *Proceedings of HLT/NAACL*.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

K. Crammer, O. Dekel, J. Keshet, S.Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:581–585.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING 1996*, pages 340–345.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald, K. Cramer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Human Language Technology Conf. and the Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

R. McDonald, K. Crammer, and F. Pereira. 2006. Multilingual dependency parsing with a two-stage discriminative parser. In *Conference on Natural Language Learning*.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and

R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL)*, pages 221–225.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

B. Taskar, D. Klein, M. Collins, D. Koller, and C.D. Manning. 2004. Max-margin parsing. In *proceedings of EMNLP*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2005. Support vector machine learning for interdependent and structured output spaces. In *proceedings ICML 2004*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Global Learning of Labelled Dependency Trees

Michael Schiehlen **Kristina Spranger**
Institute for Computational Linguistics
University of Stuttgart
D-70174 Stuttgart
Michael.Schiehlen@ims.uni-stuttgart.de
Kristina.Spranger@ims.uni-stuttgart.de

Abstract

In the paper we describe a dependency parser that uses exact search and global learning (Crammer et al., 2006) to produce labelled dependency trees. Our system integrates the task of learning tree structure and learning labels in one step, using the same set of features for both tasks. During label prediction, the system automatically selects for each feature an appropriate level of smoothing. We report on several experiments that we conducted with our system. In the shared task evaluation, it scored better than average.

1 Introduction

Dependency parsing is a topic that has engendered increasing interest in recent years. One promising approach is based on exact search and structural learning (McDonald et al., 2005; McDonald and Pereira, 2006). In this work we also pursue this approach. Our system makes no provisions for non-projective edges. In contrast to previous work, we aim to learn labelled dependency trees at one fell swoop. This is done by maintaining several copies of feature vectors that capture the features' impact on predicting different dependency relations (deprels). In order to preserve the strength of McDonald et al. (2005)'s approach in terms of unlabelled attachment score, we add feature vectors for generalizations over deprels. We also employ various reversible transformations to reach treebank formats that better match our feature representation and that reduce the complexity of the learning task. The paper first presents the methodology used, goes on to describing experiments and results and finally concludes.

2 Methodology

2.1 Parsing Algorithm

In our approach, we adopt Eisner (1996)'s bottom-up chart-parsing algorithm in McDonald et al. (2005)'s formulation, which finds the best projective dependency tree for an input string $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$. We assume that every possible head–dependent pair i, j is described by a feature vector $\boldsymbol{\Phi}_{ij}$ with associated weights $\boldsymbol{w}_{ij}$. Eisner's algorithm achieves optimal tree packing by storing partial structures in two matrices ⏢ and ◺. First the diagonals of the matrices are initiated with 0; then all other cells are filled according to eqs. (1) and (2) and their symmetric variants.

$$
\begin{aligned}
{}^{i}\searrow_{j} &= \boldsymbol{w}_{ij} \cdot \boldsymbol{\Phi}_{ij} \\
{}_{i}\text{⏢}_{j} &= \max_{k:i \leq k < j} {}_{i}\text{◺}_{k} + {}_{k+1}\text{◿}_{j} + {}^{i}\searrow_{j} && (1) \\
{}_{i}\text{◺}_{j} &= \max_{k:i < k \leq j} {}_{i}\text{⏢}_{k} + {}_{k}\text{◺}_{j} && (2) \\
\text{root} &= \max_{k:1 \leq k \leq n} {}_{1}\text{◿}_{k} + {}_{k}\text{◺}_{n} + \boldsymbol{w}_{0k} \cdot \boldsymbol{\Phi}_{0k}
\end{aligned}
$$

This algorithm only accommodates features for single links in the dependency graph. We also investigated an extension, McDonald and Pereira (2006)'s *second-order model*, where more of the parsing history is taken into account, viz. the last dependent k assigned to a head i. In the extended model, ⏢ is updated as defined in eq. (3); optimal packing requires a third matrix $\boxed{\vee}$.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1156–1160,
Prague, June 2007.

$$
\begin{aligned}
{}_{i}\text{⌂}_{j} &= \max_{k:i\leq k<j} \left\{ \begin{array}{c} {}_{i+1}\text{◿}_{j} \quad \text{if } k=i \\ {}_{i}\text{⌂}_{k} + {}_{k}\text{M}_{j} \text{ else} \end{array} \right\} \\
&\quad \boldsymbol{w}_{ijk} \cdot \boldsymbol{\Phi}_{ijk} \qquad (3) \\
{}_{i}\text{M}_{j} &= \max_{k:i\leq k<j} {}_{i}\text{◺}_{k} + {}_{k+1}\text{◿}_{j}
\end{aligned}
$$

2.2 Feature Representation

In deriving features, we used all information given in the treebanks, i.e. words (*w*), fine-grained POS tags (*fp*), combinations of lemmas and coarse-grained POS tags (*lcp*), and whether two tokens agree[1] (*agr* = yes, no, don't know). We essentially employ the same set of features as McDonald et al. (2005): $\boldsymbol{\phi}'_{ij} = \{w_i,\ fp_i,\ lcp_i,\ w_j,\ fp_j,\ lcp_j,\ w_i w_j,\ w_i lcp_j,\ lcp_i w_j,\ lcp_i lcp_j,\ fp_i lcp_j,\ fp_i fp_j,\ fp_i fp_j agr_{ij},\ fp_{i-1} fp_i fp_{j-1} fp_j,\ fp_{i-1} fp_i fp_j fp_{j+1},\ fp_i fp_{i+1} fp_{j-1} fp_j,\ fp_i fp_{i+1} fp_j fp_{j+1}\}$, and token features for root words $\boldsymbol{\phi}_{0r} = \{w_r, fp_r, lcp_r\}$. In the first order model, we recorded the tag of each token m between i and j ($\boldsymbol{\phi}_{ij} = \boldsymbol{\phi}'_{ij} \cup \{fp_i fp_j fp_m\}$); in the second order model, we only conditioned on the previous dependent k ($\boldsymbol{\phi}_{ij} = \boldsymbol{\phi}'_{ij} \cup \{fp_i fp_j fp_k,\ lcp_i fp_j fp_k,\ w_i fp_j fp_k\}$). All features but unary token features were optionally extended with direction of dependency ($i < j$ or $i > j$) and binned token distance ($|i - j| = 1, 2, 3, 4, \geq 5, \geq 10$).

2.3 Structural Learning

For determining feature weights $\boldsymbol{w}$, we used online passive–aggressive learning (OPAL) (Crammer et al., 2006). OPAL iterates repeatedly over all training instances $\boldsymbol{x}$, adapting weights after each parse. It tries to change weights as little as possible (*passiveness*), while ensuring that (1) the correct tree $\boldsymbol{y}$ gets at least as much weight as the best parse tree $\hat{\boldsymbol{y}}$ and (2) the difference in weight between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ rises with the average number of errors in $\hat{\boldsymbol{y}}$ (*aggressiveness*). This optimization problem has a closed–form solution:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \tau_t(\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}))$$

where

$$\tau_t = \frac{\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) - \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) + \sqrt{1 - \mathrm{LAS}(\boldsymbol{y}, \hat{\boldsymbol{y}})}}{\|\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})\|^2}$$

model order	# of features	min. per iteration	LAS
1	327,743	13.6	78.62
1	601.125	19.5	78.87
1	1,168,609	38.7	79.03
1	12,948,376 (513,611)	120.0 (13.3)	79.53
2	758,433	17.8	78.12
2	1,534,484	25.1	78.40
2	3,257,012 (181,303)	50.0 (9.8)	78.92
2	26,088,102 (582,907)	373.0 (23.5)	79.26

Table 1: Performance on devset of Italian treebank. In parentheses: reduction to non-null features after first iteration.

Having a closed–form solution, OPAL is easier to implement and more efficient than the MIRA algorithm used by McDonald et al. (2005), although it achieves a performance comparable to MIRA's on many problems (Crammer et al., 2006).

2.4 Learning Labels for Dependency Relations

So far, the presented system, which follows closely the approach of McDonald et al. (2005), only predicts unlabelled dependency trees. To derive a labeling, we departed from their approach: We split each feature along the deprel label dimension, so that each deprel d is associated with its own feature vector (cf. eq. (4), where $\otimes$ is the tensor product and Λ^o the orthogonal encoding).

$$\Phi_{ij,\Lambda^o(d)} = \phi_{ij} \otimes \Lambda^o(d) \qquad (4)$$

In parsing, we only consider the best deprel label.

$$^{i}\searrow_{j} = \max_{d \in \mathcal{D}} w_{ij,\Lambda^o(d)} \Phi_{ij,\Lambda^o(d)} \qquad (5)$$

On its own, this simple approach led to a severe degradation of performance, so we took a step back by re-introducing features for unlabelled trees. For each set of deprels $\mathcal{D}$, we designed a taxonomy $\mathcal{T}$ with a single maximal element (complete abstraction over deprel labels) and one minimal element for each deprel label. We also included an intermediate layer in $\mathcal{T}$ that collects *classes* of deprels, such as

[1] Agreement was computed from morphological features, viz. gender, number and person, and case. In languages with subject–verb agreement, we added a nominative case feature to finite verbs. In Basque, agreement is case-specific (absolutive, dative, ergative, other case).

Language	# tokens			DevTest	# of	min. per
	Train	DevTest	Test	Split	Features	Cycle
Catalan	425,915	4,929	5,016	89–1	3,055,518	575.0
Basque	48,019	2,507	5,390	19–1	1,837,155	37.4
Turkish	61,951	3,231	4,513	19–1	1,412,000	26.1
English	441,333	5,240	5,003	86–1	3,609,671	727.2
Greek	62,137	3,282	4,804	19–1	2,723,891	58.0
Hungarian	123,266	8,533	7,344	15–1	2,583,593	148.2
Czech	427,338	4,958	4,724	88–1	1,971,599	591.6
Chinese	333,148	4,027	5,161	82–1	1,672,360	1,015.2
Italian	67,593	3,606	5,096	19–1	1,534,485	52.0
Arabic	107,804	3,865	5,124	27–1	1,763,063	110.0

Table 2: Figures for Experiments on Treebanks.

complement, adjunct, marker, punctuation, or coordination deprels, and in this way provides for better smoothing. The taxonomy translates to an encoding Λ^t, where $\lambda_i^t(d) = 1$ iff node i in $\mathcal{T}$ is an ancestor of d (Tsochantaridis et al., 2004). Substituting Λ^t for Λ^o leads to a massive amount of features, so we pruned the taxonomy on a feature–to–feature basis by merging all nodes on a level that only encompass deprels that never occur with this feature in the training data.

2.5 Treebank Transformations

Having no explicit feature representation for the information in the morphological features slot (cf. section 2.2), we partially redistributed that information to other slots: Verb form, case[2] to *fp*, semantic classification to an empty lemma slot (Turkish affixes, e.g. "Able", "Ly"). The balance between *fp* and *w* was not always optimal; we used a fine-grained[3] classification in punctuation tags, distinguished between prepositions (e.g. *in*) and preposition–article combinations (e.g. *nel*) in Italian[4] on the basis of number/gender features, and collected definite and indefinite articles under one common *fp* tag.

When distinctions in deprels are recoverable from context, we removed them: The dichotomy between conjunctive and disjunctive coordination in Italian depends in most cases exclusively on the coordinating conjunction. The Greek and Czech treebanks have a generic distinction between ordinary deprels and deprels in a coordination, apposition, and parenthesis construction. In Greek, we got rid of the parenthesis markers on deprels by switching head and dependent, giving the former head (the parenthesis) a unique new deprel. For Czech, we reduced the number of deprels from 46 to 34 by swapping the deprels of conjuncts, appositions, etc. and their heads (coordination or comma). Sometimes, multiple conjuncts take different deprels. We only provided for the clash between "ExD" (ellipsis) and other deprels, in which case we added "ExD", see below.

1	Minimálně	3	AuxZ		
2	dva	3	Atr		
3	stupně	0	ExD		
4	rozlišení	5	Atr_M	⇒	-Apos
5	-	3	Apos	⇒	Atr
6	standard	7	ExD_M	⇒	-Coord
7	a	5	Coord_M	⇒	-Apos:ExD
8	jemně	7	ExD_M	⇒	-Coord
9	.	0	AuxK		

In Basque, agreement is usually between arguments and *auxiliary* verbs, so we re-attached[5] relevant arguments from main verb to auxiliary verb.

The training set for Arabic contains some very long sentences (up to 396 tokens). Since context-free parsing sentences of this length is tedious, we split up all sentences at final punctuation signs

[2] Case was transferred to *fp* only if important for determination of deprel (CA, HU, IT).

[3] Classes of punctuation are e.g. opening and closing brackets, commas and punctuation signalling the end of a sentence.

[4] Prep and PrepArt behave differently syntactically (e.g. an article can only follow a genuine preposition).

[5] Unfortunately, we did not take into account projectivity, so this step resulted in a steep increase of non-projective edges (9.4% of all edges) and a corresponding degradation of our evaluation results in Basque.

Language	LAS			UAS			LAcc		
	Dev	Test	AV	Dev	Test	AV	Dev	Test	AV
Basque	68.85	66.75	**68.06**	74.59	73.25	**75.15**	78.82	**76.64**	76.06
Greek	73.49	**72.29**	70.22	82.08	**80.47**	77.78	84.19	**83.16**	81.26
Turkish	70.30	72.48	**73.19**	77.97	79.33	**80.33**	81.67	82.18	**82.29**
Italian	78.23	**80.46**	78.06	82.50	**84.54**	82.45	86.30	**87.44**	85.75
Arabic	69.26	**70.08**	68.34	79.61	**81.07**	78.84	82.25	**82.32**	81.79
Hungarian	74.29	**73.90**	71.49	78.69	**78.61**	76.34	87.82	**87.60**	85.89
Chinese	84.06	**80.04**	76.59	88.25	**85.45**	81.98	87.04	**83.28**	80.16
Catalan	85.17	**85.75**	79.85	90.04	**90.79**	87.98	91.13	**91.29**	86.32
Czech	73.26	**73.86**	70.12	81.63	**81.73**	77.56	81.36	**82.03**	79.66
English	86.93	**86.21**	80.95	88.45	**88.91**	82.67	91.97	**90.89**	87.69
Basque (rev.)	72.32	**70.48**	68.06	77.78	**76.72**	75.15	80.57	**78.85**	76.06
Turkish (rev.)	74.50	**76.31**	73.19	81.12	**82.76**	80.33	84.90	**85.46**	82.29

Table 3: Results on DevTest and Test Sets compared with the Average Performance in CoNLL'07. LAS = Labelled Attachment Score, UAS = Unlabelled Attachment Score, LAcc = Label Accuracy, AV = Average score.

(AuxK). With this trick, we pushed down maximal sentence length to 196.

Unfortunately, we overlooked the fact that in Turkish, the ROOT deprel not only designates root nodes but also attaches some punctuation marks. This often leads to non-projective structures, which our parser cannot handle, so our parser scored below average in Turkish. In after–deadline experiments, we took this feature of the Turkish treebank into account and achieved above–average results by re-linking all ROOT-ed punctuation signs to the immediately preceding token.

3 Experiments and Results

All experiments were conducted on the treebanks provided in the shared task (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003). For our contribution, we used the second-order algorithm; only afterwards did we also apply the first-order model to the data, with quite good results (cf. Table 1). For testing our approach, we split the treebanks provided into an actual training and a development set (details are in Table 2). From each training set, we extracted at least a million features (not counting the split for deprel labels). The last column in Table 2 shows the average time needed in a training iteration.

For nearly all languages, our approach achieved a performance better than average (see Table 3). Only in Turkish and Basque did we score below average. On closer inspection, we saw that this performance was due to our projectivity assumption and to insufficient exploration of these treebanks. In its bottom part, Table 3 gives results of improved versions of our approach.

4 Conclusion

We presented an approach to dependency parsing that is based on exact search and global learning. Special emphasis is laid on an integrated derivation of labelled and unlabelled dependency trees. We also employed various transformation techniques to reach treebank formats that are better suited to our approach. The approach scores better than average in (nearly) all languages. Nevertheless, it is still a long way from cutting–edge performance. One direction we would like to explore in the future is the integration of dynamic features on deprel labels.

Acknowledgements

We would like to thank the organizing team for making possible again a great shared task at CoNLL!

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive–Aggressive Algorithms. *Journal of Machine Learning*, 7:551–585.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, Copenhagen, Denmark.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, Trento, Italy.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *Proceedings of the 21st International Conference on Machine Learning*.

Pro3Gres Parser in the CoNLL Domain Adaptation Shared Task

Gerold Schneider and **Kaarel Kaljurand** and **Fabio Rinaldi** and **Tobias Kuhn**
Institute of Computational Linguistics, University of Zurich
Binzmühlestrasse 14
CH - 8050 Zurich, Switzerland
{gschneid,kalju,rinaldi,tkuhn}@ifi.uzh.ch

Abstract

We present Pro3Gres, a deep-syntactic, fast dependency parser that combines a hand-written competence grammar with probabilistic performance disambiguation and that has been used in the biomedical domain. We discuss its performance in the domain adaptation open submission. We achieve average results, which is partly due to difficulties in mapping to the dependency representation used for the shared task.

1 Introduction

The Pro3Gres parser is a dependency parser that combines a hand-written grammar with probabilistic disambiguation. It is described in detail in (Schneider, 2007). It uses tagger and chunker pre-processors – parsing proper happens only between heads of chunks – and a post-processor graph converter to capture long-distance dependencies. Pro3Gres is embedded in a flexible XML pipeline. It has been applied to many tasks, such as parsing biomedical literature (Rinaldi et al., 2006; Rinaldi et al., 2007) and the whole British National Corpus, and has been evaluated in several ways. We have achieved average results in the CoNLL domain adaptation track open submission (Marcus et al., 1993; Johansson and Nugues, 2007; Kulick et al., 2004; MacWhinney, 2000; Brown, 1973). The performance of the parser is seriously affected by mapping problems to the particular dependency representation used in the shared task.

The paper is structured as follows. We give a brief overview of the parser and its design policy in section 2, we describe the domain adaptations that we have used in section 3, comment on the results obtained in section 4 and conclude in section 5.

2 Pro3Gres and its Design Policy

There has been growing interest in exploring the space between Treebank-trained probabilistic grammars (e.g. (Collins, 1999; Nivre, 2006)) and formal grammar-based parsers integrating statistics (e.g. (Miyao et al., 2005; Riezler et al., 2002)). We have developed a parsing system that explores this space, in the vein of systems like (Kaplan et al., 2004), using a linguistic *competence* grammar and a probabilistic *performance* disambiguation allowing us to explore interactions between lexicon and grammar (Sinclair, 1996). The parser has been explicitly designed to be deep-syntactic like a formal grammar-based parser, by using a dependency representation that is close to LFG f-structure, but at the same time mostly context-free and integrating shallow approaches and aggressive pruning in order to keep search-spaces small, without permitting compromise on performance or linguistic adequacy. (Abney, 1995) establishes the chunks and dependencies model as a well-motivated linguistic theory. The non-local linguistic constraints that a hand-written grammar allows us to formulate, e.g. expressing X-bar principles or barring very marked constructions, further reduce parsing time by at least an order of magnitude. Since the grammar is on Penn tags (except for few closed classed words, e.g. allowing *including* to function as preposition) the effort for writing it manually is manageable. It has been developed from scratch in about a person month,

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1161–1165,
Prague, June 2007.

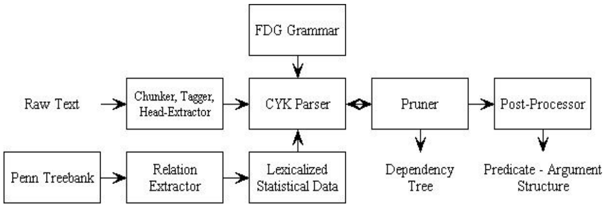


Figure 1: Pro3Gres parser flowchart

using traditional grammar engineering development cycles. It contains about 1000 rules, the number is largely so high due to tag combinatorics: for example, the various subject attachment rules combining a subject (*_NN, _NNS, _NNP, _NNPS*) and a verb (*_VBZ, _VBP, _VBG, _VBN, _VBD*) are all very similar.

The parser is fast enough for large-scale application to unrestricted texts, and it delivers dependency relations which are a suitable base for a range of applications. We have used it to parse the entire 100 million words British National Corpus (http://www.natcorp.ox.ac.uk) and similar amounts of biomedical texts. Its parsing speed is about 500,000 words per hour. The flowchart of the parser can be seen in figure 1.

Pro3Gres (PRObabilistic PROlog-implemented RObust Grammatical Role Extraction System) uses a dependency representation that is close to LFG f-structure, in order to give it an established linguistic background. It uses post-processing graph structure conversions and mild context-sensitivity to capture long-distance dependencies. We have argued in (Schneider, 2005) that LFG f-structures can be parsed for in a completely context-free fashion, except for embedded WH-questions, where a device such as functional uncertainty (Kaplan and Zaenen, 1989) or the equivalent Tree-Adjoining Grammar Adjoining operation (Joshi and Vijay-Shanker, 1989) is used. In Dependency Grammar, this device is also known as *lifting* (Kahane et al., 1998; Nivre and Nilsson, 2005).

We use a hand-written competence grammar, combined with performance-driven disambiguation obtained from the Penn Treebank (Marcus et al., 1993). The Maximum-Likelihood Estimation (MLE) probability of generating a dependency relation R given lexical heads (a and b) at distance (in chunks) δ is calculated as follows.

$$p(R,\delta|a,b) \cong p(R|a,b) \cdot p(\delta|R) = \frac{\#(R,a,b)}{\sum_{i=1}^{n} \#(R_i,a,b)} \cdot \frac{\#(R,\delta)}{\#R}$$

The counts are backed off (Collins, 1999; Merlo and Esteve Ferrer, 2006). The backoff levels include semantic classes from WordNet (Fellbaum, 1998): we back off to the lexicographer file ID of the most frequent word sense. An example output of the parser is shown in figure 2.

3 Domain Adaptation

Based on our experience with parsing texts form the biomedical domain, we have used the following two adaptations to the domain of chemistry.

(Hindle and Rooth, 1993) exploit the fact that in sentence-initial *NP PP* sequences the PP unambiguously attaches to the noun. We have observed that in sentence-initial *NP PP PP* sequences, also the second PP frequently attaches to the noun, the noun itself often being a relational noun. We have thus used such sequences to learn relational nouns from the unlabelled domain texts. Relational nouns are allowed to attach several argument PPs in the grammar, all other nouns are not.

Multi-word terms, adjective-preposition constructions and frequent PP-arguments have strong collocational force. We have thus used the collocation extraction tool XTRACT (Smadja, 2003) to discover collocations from large domain corpora. The probability of generating a dependency relation is augmented for collocations above a certain threshold. Since the tagging quality of the Chemistry testset is high, the impact of multi-word term recognition was lower than the biomedical domain when using a standard tagger, as we have shown in (Rinaldi et al., 2007).

For the CHILDES domain, we have not used any adaptation. The hand-written grammar fares quite well on most types of questions, which are very frequent in this domain. In the spirit of the shared task, we have not attempted to correct tagging errors, which were frequent in the CHILDES domain. We have restricted the use of external resources to the hand-written, domain-independent grammar, and to WordNet. Due to serious problems in mapping our

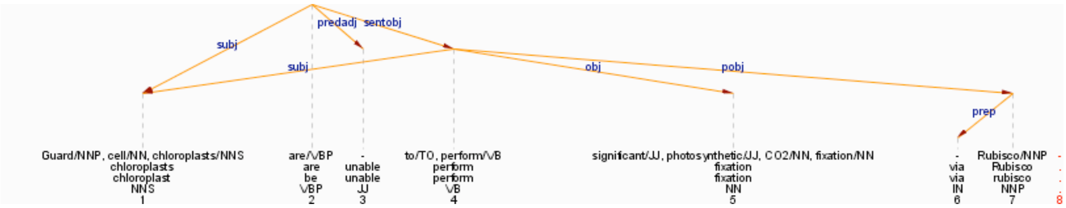


Figure 2: Example of original parser output

LFG f-structure based dependencies to the CoNLL representation, much less time than expected was available for the domain adaptation.

4 Our Results

We have achieved average results: Labeled attachment score: 3151 / 5001 * 100 = 63.01, unlabeled attachment score: 3327 / 5001 * 100 = 66.53, label accuracy score: 3832 / 5001 * 100 = 76.62. These results are about 10 % below what we typically obtain when using our own dependency representation or GREVAL (Carroll et al., 2003), a deep-syntactic annotation scheme that is close to ours. Detailed evaluations are reported in (Schneider, 2007). Our mapping was quite poor, especially when conjunctions are involved. Also punctuation is attached poorly. 5.7 % of all dependencies remained unmapped (*unknown* in the figure). We give an overview of the the relation-dependent results in figures 1 and 2.

Mapping problems include the following examples. First, headedness is handled very differently: while we assume auxiliaries, prepositions and coordinations to be dependents, the CoNNL representation assumes the opposite, which leads to incorrect mapping under complex interactions. Second, the semantics of parentheticals (*PRN*) partly remains unclear. In *Quinidine elimination was capacity limited with apparent Michaelis constant (appKM) of 2.6 microM (about 1.2 mg/L)* the gold standard annotates the second parenthesis as parenthetical, but the first as nominal modification, although both may be said to have appositional character. Third, we seem to have misinterpreted the roles of *ADV* and *AMOD*, as they are often mutually exchanged. Fourth, the logical subject (*LGS*) is sometimes marked on the by-PP (... *are strongly inhibited by-LGS carbon monoxide*) and sometimes on the participle (... *are increased-LGS by pretreatment*) in the gold standard. Relations between heads of chunks, which are central for predicate-argument structures which Pro3Gres aims to recover, such as *SBJ, NMOD, ROOT*, perform better than those for which Pro3Gres was not originally designed, particularly *ADV, AMOD, PRN, P*. Performance on *COORD* was particularly disappointing. Generally, mapping problems between different representations would be smaller if one used a dependency representation that maximally abstracts away from form to function, for example (Carroll et al., 2003).

deprel	gold	correct	system	recall (%)	prec. (%)
ADV	366	212	302	57.92	70.20
AMOD	87	8	87	9.20	9.20
CC	11	0	0	0.00	NaN
COORD	402	233	342	57.96	68.13
DEP	9	0	0	0.00	NaN
EXP	2	0	0	0.00	NaN
GAP	14	0	0	0.00	NaN
IOBJ	3	0	0	0.00	NaN
LGS	37	0	0	0.00	NaN
NMOD	1813	1576	1763	86.93	89.39
OBJ	185	146	208	78.92	70.19
P	587	524	525	89.27	99.81
PMOD	681	533	648	78.27	82.25
PRN	34	13	68	38.24	19.12
ROOT	195	138	190	70.77	72.63
SBJ	279	217	296	77.78	73.31
VC	129	116	136	89.92	85.29
VMOD	167	116	149	69.46	77.85
unknown	0	0	287	NaN	0.00

Table 1: Prec.&recall of DEPREL

We have obtained results slightly above average on the CHILDES domain, although we did not adapt the parser to this domain in any way (unlabeled attachment score: 3013 / 4999 * 100 = 60.27 %). The hand-written grammar, which includes rules for most types of questions, fares relatively well on this domain since questions are rare in the Penn Treebank (see (Hermjakob, 2001)). Pro3Gres has been employed for question parsing at a TREC conference (Burger and Bayer, 2005).

deprel	gold	correct	system	recall (%)	prec. (%)
ADV	366	161	302	43.99	53.31
AMOD	87	5	87	5.75	5.75
CC	11	0	0	0.00	NaN
COORD	402	170	342	42.29	49.71
DEP	9	0	0	0.00	NaN
EXP	2	0	0	0.00	NaN
GAP	14	0	0	0.00	NaN
IOBJ	3	0	0	0.00	NaN
LGS	37	0	0	0.00	NaN
NMOD	1813	1392	1763	76.78	78.96
OBJ	185	140	208	75.68	67.31
P	587	221	525	37.65	42.10
PMOD	681	521	648	76.51	80.40
PRN	34	12	68	35.29	17.65
ROOT	195	138	190	70.77	72.63
SBJ	279	190	296	68.10	64.19
VC	129	116	136	89.92	85.29
VMOD	167	85	149	50.90	57.05
unknown	0	0	287	NaN	0.00

Table 2: Prec.&recall of DEPREL+ATTACHMENT

5 Conclusion

We have described the Pro3Gres parser. We have achieved average results in the shared task with relatively little adaptation. Mapping to different representations is an often underestimated task. Our performance on the CHILDES task, where we did not adapt the parser, indicates that hand-written, carefully engineered *competence* grammars may be relatively domain-independent while *performance* disambiguation is more domain-dependent. We will adapt the parser to further domains and include more unsupervised learning methods.

References

Steven Abney. 1995. Chunks and dependencies: Bringing processing evidence to bear on syntax. In Jennifer Cole, Georgia Green, and Jerry Morgan, editors, *Computational Linguistics and the Foundations of Linguistic Theory*, pages 145–164. CSLI.

R. Brown. 1973. *A First Language: The Early Stages*. Harvard University Press.

John D. Burger and Sam Bayer. 2005. MITRE's Qanda at TREC-14. In E. M. Voorhees and Lori P. Buckland, editors, *The Fourteenth Text REtrieval Conference (TREC 2005) Notebook*.

John Carroll, Guido Minnen, and Edward Briscoe. 2003. Parser evaluation: using a grammatical relation annotation scheme. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 299–316. Kluwer, Dordrecht.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Ulf Hermjakob. 2001. Parsing and question classification for question answering. In *Proceedings of the ACL 2001 Workshop on Open-Domain Question Answering*, Toulouse, France.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19:103–120.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

Aravind K. Joshi and K. Vijay-Shanker. 1989. Treatment of long-distance dependencies in LFG and TAG: Functional uncertainty in LFG is a corollary in TAG. In *Proceedings of ACL '89*.

Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *Proceedings of COLINGACL*, volume 1, pages 646–652, Montreal.

Ronald Kaplan and Annie Zaenen. 1989. Long-distance dependencies, constituent structure, and functional uncertainty. In Mark Baltin and Anthony Kroch, editors, *Alternative Concepts of Phrase Structrue*, pages 17 – 42. Chicago University Press.

Ron Kaplan, Stefan Riezler, Tracy H. King, John T. Maxwell III, Alex Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of HLT/NAACL 2004*, Boston, MA.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

B. MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Paola Merlo and Eva Esteve Ferrer. 2006. The notion of argument in PP attachment. *Computational Linguistics*, 32(2):341 – 378.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2005. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In Keh-Yih Su, Jun'ichi Tsujii, Jong-Hyeok Lee, and Oi Yee Kwong, editors, *Natural Language Processing - IJCNLP 2004*, pages 684–693. Springer.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Joakim Nivre. 2006. Constraints on non-projective dependency parsing. In *Proceedings of the European Chapter of the Association of Computational Linguistics (EACL) 2006*, pages 73 – 80, Trento, Italy. Association for Computational Linguistics.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadephia, PA.

Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand, Michael Hess, and Martin Romacker. 2006. . an environment for relation mining over richly annotated corpora: the case of GENIA. *BMC Bioinformatics*, 7(Suppl 3):S3.

Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand, Michael Hess, Christos Andronis, Ourania Konstanti, and Andreas Persidis. 2007. Mining of functional relations between genes and proteins over biomedical scientific literature using a deep-linguistic approach. *Journal of Artificial Intelligence in Medicine*, 39:127 – 136.

Gerold Schneider. 2005. A broad-coverage, representationally minimal LFG parser: chunks and F-structures are sufficient. In Mriram Butt and Traci Holloway King, editors, *The 10th international LFG Conference (LFG 2005)*, Bergen, Norway. CSLI.

Gerold Schneider. 2007. *Hybrid Long-Distance Functional Dependency Parsing*. Doctoral Thesis, Institute of Computational Linguistics, University of Zurich. accepted for publication.

John Sinclair. 1996. The empty lexicon. *International Journal of Corpus Linguistics*, 1, 1996.

Frank Smadja. 2003. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19:1, Special issue on using large corpora:143–177.

Structural Correspondence Learning for Dependency Parsing

Nobuyuki Shimizu
Information Technology Center
University of Tokyo
Tokyo, Japan
shimizu@r.dl.itc.u-tokyo.ac.jp

Hiroshi Nakagawa
Information Technology Center
University of Tokyo
Tokyo, Japan
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract

Following (Blitzer et al., 2006), we present an application of structural correspondence learning to non-projective dependency parsing (McDonald et al., 2005). To induce the correspondences among dependency edges from different domains, we looked at every two tokens in a sentence and examined whether or not there is a preposition, a determiner or a helping verb between them. Three binary linear classifiers were trained to predict the existence of a preposition, etc, on unlabeled data and we used singular value decomposition to induce new features. During the training, the parser was trained with these additional features in addition to these described in (McDonald et al., 2005). We discriminatively trained our parser in an on-line fashion using a variant of the voted perceptron (Collins, 2002; Collins and Roark, 2004; Crammer and Singer, 2003).

1 Introduction

We have recently seen growing popularity of dependency parsing. It is no longer rare to see dependency relations used as features, in tasks such as machine translation (Ding and Palmer, 2005) and relation extraction (Bunescu and Mooney, 2005). However, there is one factor that prevents the use of dependency parsing: sparseness of annotated corpora outside Wall Street Journal. In many situations we need to parse sentences from a target domain with no labeled data, which is a different distribution from a source domain where plentiful labeled training data is available.

In this paper, we investigate the effectiveness of structural correspondence learning (SCL) (Blitzer et al., 2006) in the domain adaptation task given by the CoNLL 2007. They hypothesize that a model trained in the source domain using this common feature representation will generalize better to the target domain, and focus on using unlabeled data from both the source and target domains to learn a common feature representation that is meaningful across both domains.

The paper is structured as follows: in section 2, we review the decoding and learning aspects of (McDonald et al., 2005), in section 3, structural correspondence learning applied to dependency parsing, and in section 4, we describe the experiments and the features needed for the CoNLL 2006 shared task.

2 Non-Projective Dependency Parsing

2.1 Dependency Structure

Let us define x to be a generic sequence of input tokens together with their POS tags and other morphological features, and y to be a generic dependency structure, that is, a set of edges for x.

A labeled edge is a tuple $\langle \textit{DEPREL}, i \rightarrow j \rangle$ where i is the start point of the edge, j is the end point, and *DEPREL* is the label of the edge. The token at i is the head of the token at j.

Table 1 shows our formulation of a structured prediction problem. Given x, the input tokens and their features (column 2 and 3, Table 1), the task is to pre-

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1166–1169,
Prague, June 2007. ©2007 Association for Computational Linguistics

Index	Token	POS	Labeled Edge
1	John	NN	$\langle SUBJ, 2 \rightarrow 1 \rangle$
2	saw	VBD	$\langle PRED, 0 \rightarrow 2 \rangle$
3	a	DT	$\langle DET, 4 \rightarrow 3 \rangle$
4	dog	NN	$\langle OBJ, 2 \rightarrow 4 \rangle$
5	yesterday	RB	$\langle ADJU, 2 \rightarrow 5 \rangle$
6	which	WDT	$\langle MODWH, 7 \rightarrow 6 \rangle$
7	was	VBD	$\langle MODPRED, 4 \rightarrow 7 \rangle$
8	a	DT	$\langle DET, 10 \rightarrow 8 \rangle$
9	Yorkshire	NN	$\langle MODN, 10 \rightarrow 9 \rangle$
10	Terrier	NN	$\langle OBJ, 7 \rightarrow 10 \rangle$
11	.	.	$\langle ., 10 \rightarrow 11 \rangle$

Table 1: Example Edges

dict y, the set of labeled edges (column 4, Table 1).

In this paper we use the common method of factoring the score of the dependency structure as the sum of the scores of all the labeled edges. A dependency structure is characterized by its labeled edges, and for each labeled edge, we have features and corresponding weights. The score of a dependency structure is the sum of these weights.

For example, let us say we would like to find the score of the labeled edge $\langle OBJ, 2 \rightarrow 4 \rangle$. This is the edge going to the 4th token ”dog” in Table 1. The features for this edge could be:

- There is an edge starting at saw, with the POS tag VBD, and the distance between the head and the child is 2. ($head = word_j, head_{POS} = pos_j, dist(i,j) = |i-j|$)
- There is an edge ending at dog, with the POS tag NN, and the distance between the head and the child is 2. ($child = word_i, child_{POS} = pos_i, dist(i,j) = |i-j|$)

In the upcoming section, we explain a decoding algorithm for the dependency structures, and later we give a method for learning the weight vector used in the decoding.

2.2 Maximum Spanning Tree Algorithm

As in (McDonald et al., 2005), we use Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965; Edmonds, 1967) for decoding. CLE finds the Maximum Spanning Tree in a directed graph. The following is a summary given in (McDonald et al., 2005).

> Informally, the algorithm has each vertex in the graph greedily select the incoming edge with highest weight.

Note that the edge is coming from the parent to the child. That is, given a child node $word_j$, we are finding the parent, or the head $word_i$ such that the edge (i,j) has the highest weight among all $i, i \neq j$.

> If a tree results, then this must be the maximum spanning tree. If not, there must be a cycle. The procedure identifies a cycle and contracts it into a single vertex and recalculates edge weights going into and out of the cycle. It can be shown that a maximum spanning tree on the contracted graph is equivalent to a maximum spanning tree in the original graph (Leonidas, 2003). Hence the algorithm can recursively call itself on the new graph.

2.3 Online Learning

Again following (McDonald et al., 2005), we have used the single best MIRA (Crammer and Singer, 2003), which is a “margin aware” variant of perceptron (Collins, 2002; Collins and Roark, 2004) for structured prediction. In short, the update is executed when the decoder fails to predict the correct parse, and we compare the correct parse y^t and the incorrect parse y' suggested by the decoding algorithm. The weights of the features in y' will be lowered, and the weights of the features in y^t will be increased accordingly.

3 Domain Adaptation

Following (Blitzer et al., 2006), we present an application of structural correspondence learning (SCL) to non-projective dependency parsing (McDonald et al., 2005). SCL is a method for adapting a classifier learned in a source domain to a target domain. We assume that both domains have unlabeled data, but only the source domain has labeled training data.

SCL works as follows: 1. Define a set of pivot features on the unlabeled data from both domains. 2. Use these pivot features to learn a mapping from the original feature spaces of both domains to a shared, low-dimensional real-valued feature space. A high inner product in this new space indicates a high degree of correspondence. 3. Use both the transformed and original features from the source domain. 4. Again using both the transformed and original features, test the samples from the target domain. If we learned a good mapping, then the effectiveness of the classifier in the source domain should transfer to the target domain.

To induce the correspondences among dependency edges in the source domain and the target domain, we looked at every two tokens in a sentence and examined whether or not there is a preposition, a determiner or a helping verb between them. Although no edge is present in unlabeled data, the presence of a preposition indicates that this edge between the tokens, if existed, will not be a noun modifier (in English corpus, this label is NMOD). Thus, this induced feature should correlate with the label of an edge candidate. We postulate that the label of an edge candidate, if known, may allow the supervised learner to choose the correct edge among the edge candidates in the target domain.

In the first step, we chose the presence of a preposition, a determiner or a helping verb between tokens as pivot features. Then three binary linear classifiers were trained to predict the existence of a preposition ($prep$), determiner (det) and helping verb (hv) on unlabeled data and obtained a weight vector for each classifier.

$$classifier_{prep}(e) = sign(w_{prep}\phi(e))$$
$$classifier_{det}(e) = sign(w_{det}\phi(e))$$
$$classifier_{hv}(e) = sign(w_{hv}\phi(e))$$

The input to the above classifiers is an edge e instead of a whole sentence x. ϕ is a mapping from an edge to a feature vector. Since POS tags were not available in unlabeled data, for pivot predictors, we took the subset of the features given by an edge. The features for pivot predictors are listed in Table 2. The reminder of the features are the same as ones used in (McDonald et al., 2005).

Using each weight vector as a column, we created a weight matrix. $W = [w_{prep}|w_{det}|w_{hv}]$. And run a singular value decomposition to induce a lower dimensional feature space. $W = U\Sigma V$. We then took the transpose of the resulting unitary matrix, $U^\top$ which maps the original data to the space spanned by the principal components, and applied it to the feature vector of every potential edge. The original feature vector is $\begin{pmatrix} \mathbf{f}_{subset} \\ \mathbf{f}_{reminder} \end{pmatrix}$. We argument the feature vector with the additional feature induced by $U^\top$. The augmented feature vectors $\begin{pmatrix} \mathbf{f}_{subset} \\ \mathbf{f}_{reminder} \\ U^\top \mathbf{f}_{subset} \end{pmatrix}$ were used throughout the training and testing of the dependency parser.

4 Experiments

Our experiments were conducted on CoNLL-2007 shared task domain adaptation track (Nivre et al., 2007) using treebanks (Marcus et al., 1993; Johansson and Nugues, 2007; Kulick et al., 2004).

Given an edge $\langle DEPREL, i, j\rangle$
$head_{-1} = word_{i-1}$
$head = word_i$
$head_{+1} = word_{i+1}$
$child_{-1} = word_{j-1}$
$child = word_j$
$child_{+1} = word_{j+1}$

Table 2: Binary Features for Pivot Predictors

4.1 Dependency Relation

The CLE algorithm works on a directed graph with unlabeled edges. Since the CoNLL shared task requires the labeling of edges, as a preprocessing stage, we created a directed complete graph. Then we labeled each edge with the highest scoring dependency relation. This complete graph was given to the CLE algorithm and the edge labels were never altered in the course of finding the maximum spanning tree.

4.2 Features

The features we used for pivot predictors to classify each edge $\langle DEPREL, i, j\rangle$ are shown in Table 2. The index i is the position of the parent and j is that of the child.

$word_j$ = the word token at the position j.
pos_j = the coarse part-of-speech at j.

No other features were used beyond the combinations of the word token in Table 2.

The hardware used was an Intel CPU at 3.0 Ghz with 32 GB of memory, and the software was written in C++. While more iterations should help, due to the time constraints, we were unable to complete more training. The parser required a few days to train.

5 Results

Unfortunately, we have discovered a bug in our codes after submitting our results for the blind tests, and the reported results in (Nivre et al., 2007) were not representative of our approach. The current results (closed class) are shown in Table 3.

For the explanations of Labeled Attachment Score, Unlabeled Attachment Score and Label Accuracy, the readers are suggested to refer to the shared task introductory paper (Nivre et al., 2007). WSJ represents the application of the parser without SCL to the source domain test set, and WSJ-SCL the parser with SCL to the same test set. Similarly

Domain	LAS	UAS	Label Accuracy
WSJ	83.01% → 83.43%	86.43% → 86.81%	88.77% → 88.99%
WSJ-SCL	83.43% → 83.59%	86.87% → 86.93%	88.75% → 89.01%
Chem	74.75% → 75.18%	80.74% → 81.24%	82.34% → 82.70%
Chem-SCL	75.04% → 74.91%	81.02% → 80.82%	82.18% → 82.18%

Table 3: Labeled Attachment Score, Unlabeled Attachment Score and Label Accuracy

Chem and Chem-SCL represents the application of the parser without SCL and with SCL to the source domain test set respectively. We did batch learning by running the online algorithm 4 times. An arrow → indicates how the results after 2nd iteration changed at the end of 4th iteration. Contrary to our expectations, we seem to see SCL overfitting to the source domain WSJ in this experiment. Due to the lack of POS tags in unlabeled data, our feature set for pivot predictors uses tokens extensively unlike that for the dependency parser. Since tokens are not as abstract as POS tags, we suspect induced features may have caused overfitting.

6 Conclusion

We presented an application of structural correspondence learning to non-projective dependency parsing. Effectiveness of SCL for domain adaptation is mixed in this experiment perhaps due to the mismatch between feature sets. Future work includes use of more sophisticated features such as POS and other morphological features, possibly a joint domain adaptation of POS tagging and dependency parsing for unlabeled data as well as re-examination of pivot features.

References

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*.

R. Bunescu and R. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. In *Science Sinica*, page 14:13961400.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of the 42rd Annual Meeting of the ACL*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. In *JMLR*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of the 43rd Annual Meeting of the ACL*.

J. Edmonds. 1967. Optimum branchings. In *Journal of Research of the National Bureau of Standards*, page 71B:233240.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

G. Leonidas. 2003. Arborescence optimization problems solvable by edmonds algorithm. In *Theoretical Computer Science*, page 301:427 437.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Adapting the RASP System for the CoNLL07 Domain-Adaptation Task

Rebecca Watson and Ted Briscoe
Computer Laboratory
University of Cambridge
`FirstName.LastName@cl.cam.ac.uk`

Abstract

We describe our submission to the domain adaptation track of the CoNLL07 shared task in the open class for systems using external resources. Our main finding was that it was very difficult to map from the annotation scheme used to prepare training and development data to one that could be used to effectively train and adapt the RASP system unlexicalized parse ranking model. Nevertheless, we were able to demonstrate a significant improvement in performance utilizing bootstrapping over the PBIOTB data.

1 Introduction

The CoNLL07 domain adaptation task was created to explore how a parser trained in one domain might be adapted to a new one. The training data were drawn from the PTB (Marcus *et al.*, 1993) reannotated with dependency relations (Johansson and Nugues, 2007, hereafter DRs). The test data were taken from a corpus of biomedical articles (Kulick *et al.*, 2004) and the CHILDES database (Brown, 1973; MacWhinney, 2000) also reannotated with DRs (see Nivre *et al.*, 2007) for further details of the task, annotation format, and evaluation scheme. The development data consisted of a small amount of annotated and unannotated biomedical and conversational data.

The RASP system (Briscoe *et al.*, 2006) utilizes a manually-developed grammar and outputs grammatical bilexical dependency relations (see Briscoe, 2006 for a detailed description, hereafter GRs). Watson *et al.* (2007) describe a semi-supervised, bootstrapping approach to training the parser which utilizes unlabelled partially-bracketed input with respect to the system derivations. For the domain adaptation task we retrained RASP by mapping our GR scheme to the DR scheme and annotation format, and used this mapping to select a derivation to train the unlexicalized parse ranking model from the annotated PTB training data. We also performed similar partially-supervised bootstrapping over the 200 annotated biomedical sentences in the development data. We then tried unsupervised bootstrapping from the unannotated development data based on these initial models.

As the parser requires input to consist of a sequence of one of 150 CLAWS PoS tags, we also utilize a first-order HMM PoS tagger which has been trained on manually-annotated data from the LOB, BNC and Susanne Corpora (see Briscoe, 2006 for further details). Accordingly, we submitted our results in the open class.

2 Training and Adaptation

The RASP parser is a generalized LR parser which builds a non-deterministic generalized LALR(1) parse table from the grammar (Tomita, 1987). A context-free 'backbone' is automatically derived from a unification grammar. The residue of features not incorporated into the backbone are unified on each reduce action and if unification fails the associated derivation paths also fail. The parser creates a packed parse forest represented as a graph-structured stack.

Inui *et al.* (1997) describe the probability model

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1170–1174,
Prague, June 2007. ©2007 Association for Computational Linguistics

utilized in the system where a transition is represented by the probability of moving from one stack state, σ_{i-1}, (an instance of the graph structured stack) to another, σ_i. They estimate this probability using the stack-top state s_{i-1}, next input symbol l_i and next action a_i. This probability is conditioned on the type of state s_{i-1}. S_s and S_r are mutually exclusive sets of states which represent those states reached after shift or reduce actions, respectively. The probability of an action is estimated as:

$$P(l_i, a_i, \sigma_i|\sigma_{i-1}) \approx \left\{ \begin{array}{ll} P(l_i, a_i|s_{i-1}) & s_{i-1} \in S_s \\ P(a_i|s_{i-1}, l_i) & s_{i-1} \in S_r \end{array} \right\}$$

Therefore, normalization is performed over all lookaheads for a state or over each lookahead for the state depending on whether the state is a member of S_s or S_r, respectively. In addition, Laplace estimation can be used to ensure that all actions in the table are assigned a non-zero probability.

These probabilities are estimated from counts of actions which yield derivations compatible with training data. We use a confidence-based self-training approach to select derivations compatible with the annotation of the training and development data to train the model. In Watson *et al.* (2007), we utilized unlabelled partially-bracketed training data as the starting point for this semi-supervised training process. Here we start from the DR-annotated training data, map it to GRs, and then find the one or more derivations in our grammar which yield GR output consistent with the GRs recovered from the DR scheme. Following Watson *et al.* (2007), we utilize the subset of sentences in the training data for which there is a single derivation consistent with this mapping to build an initial trained parse ranking model. Then we use this model to rank the derivations consistent with the mapping in the portion of the training data which remains ambiguous given the mapping. We then train a new model based on counts from these consistent derivations which are weighted in some manner by our confidence in them, given both the degree of remaining ambiguity and also the ranking and/or derivation probabilities provided by the initial model.

Thus, the first and hardest step was to map the DR scheme to our GR scheme. Issues concerning this mapping are discussed in section 4. Given this mapping, we determined the subset of sentences in the (PTB) training data for which there was a single derivation in the grammar compatible with the set of mapped GRs. These derivations were used to create the initial trained model (B) from the uniform model (A). To evaluate the performance of these and subsequent models, we tested them using our own GR-based evaluation scheme over 560 sentences from our reannotated version of DepBank, a subset of section 23 of the WSJ PTB (see Briscoe & Carroll, 2006). Table 1 gives the unlabelled precision, recall and microaveraged F_1 score of these models over this data. Model B was used to rerank derivations compatible with the mapped GRs recovered for the PTB training data. Model C was built from the weighted counts of actions in the initial set of unambiguous data and from the highest-ranked derivations over the training data (i.e. we do not include duplicate counts from the unambiguous data). Counts were weighted with scores ranging $[0 - 1]$ corresponding to the overall probability of the relevant derivation. The evaluation shows a steady increase in performance for these successive models. We also explored other variants of this bootstrapping approach involving use of weighted counts from the top *n* ranked parses derived from the initial model (see Watson *et al.*, 2007, for details), but none performed better than simply selecting the highest-ranked derivation.

To adapt the trained parser, we used the same technique for the 200 in-domain biomedical sentences (PBIOTB), using Model C to find the highest-ranked parse compatible with the annotation, and derived Model D from the combined counts from this and the previous training data. We then used Model D to rank the parses for the unannotated in-domain data (PBIOTB unsupervised), and derived Model E from the combined counts from the highest-ranked parses for all of the training and development data. We then iterated this process two more times over the unannotated datasets (each with an increasing number of examples though increasingly less relevant to the test data). The performance over our out-of-domain PTB-derived test data remains approximately the same for all these models. Therefore, we chose to use Model G for the blind test as it incorporates most information from the in-

Mdl.	Data	Init.	Prec.	Rec.	F_1
A	Uniform	-	71.06	69.00	70.01
	PTB				
B	Unambig.	A	75.94	73.16	74.53
C	Ambig.	B	77.88	75.11	76.47
	PBIOTB				
D	Supervised	C	77.86	75.09	76.45
E	Unsup. 1	D	77.98	75.25	76.59
F	Unsup. 2	E	77.85	75.19	76.50
G	Unsup. 3	F	77.76	75.09	76.41
	CHILDES				
H	Unsup. 1	C	78.34	75.59	76.94

Table 1: Performance of Successive Bootstrapping Models

	Score	Avg.	Std
PCHEMTB - labelled	55.47	65.11	09.64
PCHEMTB - unlab.ed	62.79	70.24	08.14
CHILDES - unlab.ed	45.61	56.12	09.17

Table 2: Official Scores

domain data. For the CHILDES data we performed one iteration of unsupervised adaptation in the same manner starting from Model C.

3 Evaluation

For the blind test submission we used Models G and H to parse the PCHEMTB and CHILDES data, respectively. We then mapped our GR output from the highest-ranked parses to the DR scheme and annotation format required by the CoNLL evaluation script. Our reported results are given in Table 2.

We used the annotated versions of the blind test data supplied after the official evaluation to assess the degree of adaptation of the parser to the in-domain data. We mapped from the DR scheme and annotation format to our GR format and used our evaluation script to calculate the precision, recall and microaveraged F_1 score for the unadapted models and their adapted counterparts on the blind test data, given in Table 3. The results for CHILDES show no evidence of adaptation to the domain. However, those for PCHEMTB show a statistically significant (Wilcoxin Signed Ranks) improvement over the initial model. The generally higher scores in Table 3, as compared to Table 2, reflect the differences between the task annotation scheme and our GR representation as well as those of the evaluation schemes, which we discuss in the next section.

Model	Test Data	Prec.	Rec.	F_1
C	PCHEMTB	71.58	73.69	72.62
G	PCHEMTB	72.32	74.56	73.42
C	CHILDES	82.64	65.18	72.88
H	CHILDES	81.71	64.58	72.14

Table 3: Performance of (Un)Adapted Models

4 Discussion

The biggest issue for us participating in the shared task was the difficulty of reconciling the DR annotation scheme with our GR scheme, given the often implicit and sometimes radical underlying differences in linguistic assumptions between the schemes.

Firstly, the PoS tagsets are different and ours contains three times the number of tags. Given that the grammar uses these tags as preterminal categories, this puts us at a disadvantage in mapping the annotated training and development data to optimal input to train the (semi-)supervised models.

Secondly, there are 17 main types of GR relation and a total of 46 distinctions when GR subtypes are taken into account – for instance the GR **ncsubj** has two subtypes depending on whether the surface subject is the underlying object of a passive clause. The DR scheme has far fewer distinctions creating similar difficulties when creating optimal (semi-)supervised training data.

Thirdly, the topology of the dependency graphs is often significantly different because of reversed head-dependent bilexical relations and their knock-on effects – for instance, the DR **AUX** relation treats the (leftmost) auxiliary as head and modifiers of the verb group attach to the leftmost auxiliary, while the GR scheme treats the main verb as (semantic) head and modifiers of the verb group attach to it.

Fourthly, the treatment of punctuation is very different. The DR scheme includes punctuation markers in DRs which attach to the root of the subgraph over which they have scope. By contrast, the GR scheme does not output punctuation marks directly but follows Nunberg's (1990) linguistic analysis of punctuation as delimiting and typing text units or adjuncts (at constituent boundaries). Thus the GR scheme includes text (adjunct) relations and treats punctuation marks as indicators of such relations – for instance, for the example *The subject GRs – ncsubj, xsubj and csubj – all have subtypes.*, RASP outputs the GR **(ta dash GRs and)** indicating that the dash-delimited parenthetical is a text adjunct of *GRs* with head *and*, while the DR scheme gives **(DEP GRs and)**, and two **(P and –)** relations corresponding to each dash mark.

Although we attempted to derive an optimal and error-free mapping between the schemes, this was hampered by the lack of information concerning the DR scheme, lack of time, and the very different approaches to punctuation. This undoubtedly limited our ability to train effectively from the PTB data and to adapt the trained parser using the in-domain data. For instance, the mean average unlabelled F_1 score between the GRs mapped from the annotated PTB training data and closest matching set of GRs output by RASP for this data is 84.56 with a standard deviation of 12.41. This means that the closest matching derivation which is used for training the initial model is on average only around 85% similar even by the unlabelled measure. Thus, the mapping procedure required to relate the annotated data to RASP derivations is introducing considerable noise into the training process.

Mapping difficulties also depressed our official scores very significantly. In training and adapting we found that bootstrapping based on unlabelled dependencies worked better in all cases than utilizing the labelled mapping we derived. For the official submission, we removed all **ta**, **quote** and **passive** GRs and mapped all punctuation marks to the **P** relation with head **0**. Furthermore, we do not generate a root relation, though we assumed any word that was not a dependent in other GRs to have the dependent **ROOT**. In our own evaluations based on mapping the annotated training and development data to our GR scheme, we remove all **P** relations and map **ROOT** relations to the type **root** which we added to our GR hierarchy. We determined the semantic head of each parse during training so as to compare against the **root** GR and better utilize this additional information. In the results given in Table 1 over our DepBank test set, the effect of removing the **P** dependencies is to depress the F_1 scores by over 20%. For the CHILDES and PCHEMTB blind test data, our F_1 scores improve by over 7% and just under 9% respectively when we factor out the effect of **P** relations. These figures give an indication of the scale of the problem caused by these representional differences.

5 Conclusions

The main conclusion that we draw from this experience is that it is very difficult to effectively relate linguistic annotations even when these are inspired by a similar (dependency-based) theoretical tradition. The scores we achieved were undoubtedly further depressed by the need to use a partially-supervised boostrapping approach to training because the DR scheme is less informative than the GR one, and by our decision to use an entirely unlexicalized parse ranking model for these experiments. Despite these difficulties, performance on the PCHEMTB dataset using the adapted model improved significantly over that of the unadapted model, suggesting that bootstrapping using confidence-based self-training is a viable technique.

Acknowledgements

This research has been partially supported by the EPSRC via the RASP project (grants GR/N36462 and GR/N36493) and the ACLEX project (grant GR/T19919). The first author is funded by the Overseas Research Students Awards Scheme and the Poynton Scholarship appointed by the Cambridge Australia Trust in collaboration with the Cambridge Commonwealth Trust.

References

E. Briscoe (2006) *An introduction to tag sequence grammars and the RASP system parser,* University of Cambridge, Computer Laboratory Technical Report, UCAM-CL-TR-662.

E. Briscoe and J. Carroll (2006) 'Evaluating the Accuracy of an Unlexicalized
Statistical Parser on the PARC DepBank', *Proceedings of the ACL-Coling'06,* Sydney, Australia.

Briscoe, E.J., J. Carroll and R. Watson (2006) ‘The Second Release of the RASP System’, *Proceedings of the ACL-Coling’06,* Sydney, Australia.

R. Brown (1973) *A First Language: The Early Stages,* Harvard University Press.

Inui, K., V. Sornlertlamvanich, H. Tanaka and T. Tokunaga (1997) ‘A new formalization of probabilistic GLR parsing’, *Proceedings of the 5th International Workshop on Parsing Technologies,* MIT, Cambridge, Massachusetts, pp. 123–134.

R. Johansson and P. Nugues (2007) *Extended Constituent-to-Dependency Conversion for English,* NODALIDA16.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein and L. Ungar (2004) ‘Integrated Annotation for Biomedical Information Extraction’, *Proceedings of the HLT-NAACL2004,* Boston, MA..

B. MacWhinney (2000) *The CHILDES Project: Tools for Analyzing Talk,* Lawrence Erlbaum.

M. Marcus, B. Santorini and M. Marcinkiewicz (1993) ‘Building a Large Annotated Corpus of English: the Penn Treebank’, *Computational Linguistics, vol.19.2,* 313–330.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel and D. Yuret (2007) ‘The CoNLL 2007 Shared Task on Dependency Parsing’, *Proceedings of the EMNLP-CoNLL2007,* Prague.

G. Nunberg (1990) *The Linguistics of Punctuation,* CSLI Publications.

Tomita, M. (1987) ‘An Efficient Augmented Context-Free Parsing Algorithm’, *Computational Linguistics, vol.13(1–2),* 31–46.

R. Watson, E. Briscoe and J. Carroll (2007) ‘Semi-supervised Training of a Statistical Parser from Unlabeled Partially-bracketed Data’, *Proceedings of the IWPT07,* Prague.

Multilingual Deterministic Dependency Parsing Framework using Modified Finite Newton Method Support Vector Machines

Yu-Chieh Wu
Dept. of Computer Science and Information Engineering
National Central University
Taoyuan, Taiwan
bcbb@db.csie.ncu.edu.tw

Jie-Chi Yang
Graduate Institute of Network Learning Technology
National Central University
Taoyuan, Taiwan
yang@cl.ncu.edu.tw

Yue-Shi Lee
Dept. of Computer Science and Information Engineering
Ming Chuan University
Taoyuan, Taiwan
lees@mcu.edu.tw

Abstract

In this paper, we present a three-step multilingual dependency parser based on a deterministic shift-reduce parsing algorithm. Different from last year, we separate the root-parsing strategy as sequential labeling task and try to link the neighbor word dependences via a near neighbor parsing. The outputs of the root and neighbor parsers were encoded as features for the shift-reduce parser. In addition, the learners we used for the two parsers and the shift-reduce parser are quite different (conditional random fields and the modified finite-Newton method support vector machines). We found that our method could benefit from the two-preprocessing stages. To speed up training, in this year, we employ the MFN-SVM (modified finite-Newton method support vector machines) which can be learned in linear time. The experimental results show that our method achieved the middle rank over the 23 teams. We expect that our method could be further improved via well-tuned parameter validations for different languages.

1 Introduction

The target of dependency parsing is to automatically recognize the head-modifier relationships between words in natural language sentences. Usually, a dependency parser can construct a similar grammar tree with the dependency graph. In this year, CoNLL-2007 shared task (Nivre et al., 2007) focuses on multilingual dependency parsing based on ten different languages (Hajic et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmova et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Czendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003) and domain adaptation for English (Marcus et al., 1993; Johansson and Nugues, 2007; Kulick et al., 2004; MacWhinney, 2000; Brown, 1973) without taking the language-specific knowledge into consideration. The ultimate goal of them is to design ideal multilingual and domain portable dependency parsing systems.

To accomplish the multilingual and domain adaptation tasks, we present a three-pass parsing model based on a shift-reducing algorithm (Yamada and Matsumoto, 2003; Chang et al., 2006), namely, neighbor parsing, root relation parsing, and shift-reduce parsing. Our method favors examining the "un-parsed" tokens, which incrementally shrink. At the beginning, the parsing direction is mainly determined by the amount of un-parsed tokens in the sentence with either forward or backward parse. In this step, the projective parsing method can be used to evaluate most of the non-projective Treebank datasets. Once the direction is determined, the pseudo-projectivize transformation algorithm (Nivre and Nilsson, 2005) converts most non-projective training data into projective and decodes the parsed text into non-projective. Hereafter, both neighbor-parser and root-parser were trained to discovery additional features for the downstream shift-reduce parse model. We found that the two additional features could improve the performance. Subsequently, the modified shift-reduce parsing algorithm starts to parse the final dependencies with two-pass processing, i.e., predict parse action and label the relations.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1175–1181,
Prague, June 2007.

In the remainder of this paper, Section 2 describes the proposed parsing model, and Section 3 lists the experimental settings and results. Section 4 presents the discussion and analysis of our parser. In Section 5, we draw the future direction and conclusion.

2 System Description

Over the past decades, many state-of-the-art parsing algorithm were proposed, such as head-word lexicalized PCFG (Collins, 1998), Maximum Entropy (Charniak, 2000), Maximum/Minimum spanning tree (MST) (McDonald et al., 2005), shift-reduce-based deterministic parsing (Yamada and Matsumoto, 2003; Chang et al., 2006; Nivre, 2003). Among them, the shift-reduce methods were shown to be the most efficient method, which only costs at most 2n~3n actions to parse a sentence (Chang et al., 2006; Nivre, 2003). Chang et al. (2006) further added the "wait-right" action to the words that had children and could not be reduced in current state. This could avoid the so-called "too early reduce" problems.

The overall parsing model can be found in Figure 1. Figure 2 illustrates the detail system spec of our parsing model.

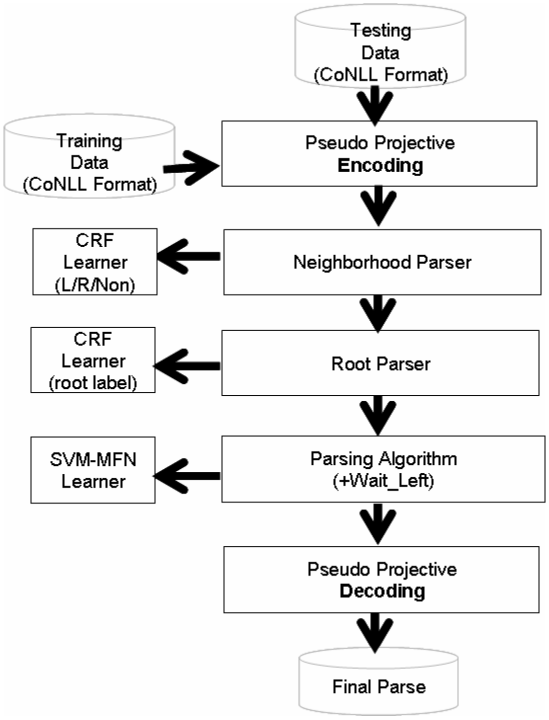

Testing Data (CoNLL Format)
Training Data (CoNLL Format)
Pseudo Projective Encoding
CRF Learner (L/R/Non)
Neighborhood Parser
CRF Learner (root label)
Root Parser
SVM-MFN Learner
Parsing Algorithm (+Wait_Left)
Pseudo Projective Decoding
Final Parse

Figure 1: System architecture

2.1 Neighbor Parser

As shown in Figure 1, the first step is to identify the neighbor head-modifier relations between two consecutive words. Cheng et al. (2006) also reported that the use of neighboring dependency attachment tagger enhance the unlabeled attachment scores from 84.38 to 84.6 for 13 languages. Usually, it is the case that the select features are fixed and could not be tuned to capture the second order features (McDonald et al., 2006). At each location, there the focus and next words are always compared. It may fail to link the next and next+1 word pair since the next word might be reduced due to an earlier wrong decision.

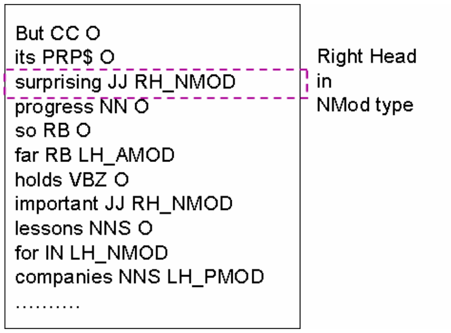
Ⅰ.	Parsing Algorithm:	1. Neighbor Parser 2. Root Parser 3. Shift-Reduce Algorithm (Yamada and Matsumoto, 2003)
Ⅱ.	Parser Characteristics:	1. Deterministic 2. two-pass (Labeling separated) 3. Pseudo-Projective en(de)-coding (Nivre and Nilsson, 2005)
Ⅲ.	Learner:	MFN-SVM (1) One-versus-All (2) Linear Kernel
Ⅳ.	Feature Set:	1. Lexical (Unigram/Bigram) 2. Fine-grained POS (and BiPOS) 3. Lemma/FEAT used
Ⅴ.	Post-Processing:	Non-Used
Ⅵ.	Additional/External Resources:	Non-Used

Figure 2: System spec

However, starting parsing based on the result of neighbor parsing is not a good idea since it could produce error propagation problems. Rather, we include the result of our neighbor parsing as features to increase the original feature set. In the preliminary study, we found that the derived features are very useful for most languages.

As conventional sequential tagging problems, such part-of-speech tagging and phrase chunking, we employ the conditional random fields (CRF) as learners (Kudo et al., 2004). The basic idea of the neighbor parsing can be shown in Figure 3.

The first and second colums in Figure 3 represents the basic word and fine-grained POS froms, while the third column indicates if this word has the LH (left-head) or RH (right-head) with associated relations or O (no neighbor head in either left or right neighbor word). The used features are:
Word, fine-grained POS, bigram, and bi-POS with context window = 2(left) and 4(right)



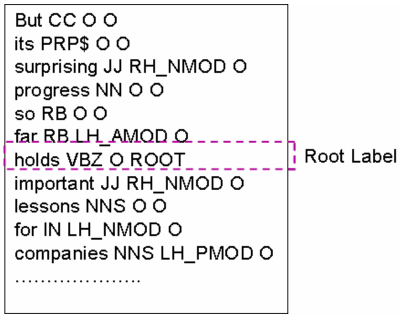
```
But CC O
its PRP$ O
surprising JJ RH_NMOD
progress NN O
so RB O
far RB LH_AMOD
holds VBZ O
important JJ RH_NMOD
lessons NNS O
for IN LH_NMOD
companies NNS LH_PMOD
……….
```

Figure 3: Sequential tagging model for neighbor parse

Unfortunately, for some languages, like Chinese and Czech, training with CRF is because of the large number of features and the head relations. To make it practical, we focus on just three types: left head, right head, and out-of-neighbor. This effectively reduces most of the feature space for the CRF. The training time for the neighbor parser with only three categories is less than 5 minutes while it takes three days with taking all the relation tag into account.

2.2 Root Parser

After the neighbor parse, the tagged labels are good features for the root parse. In the second stage, the root parser identifies the *root* words in the sentence. Nevertheless, for some languages, such as Arabic and Czech, the roots might be several types as against to Chinese and English in which the number of labels of roots is merely one. Similar to the neighbor parser, we also take the root label into account. As noted, for Chinese and English, the goal of the root parser can be reduced to determine whether the current word is root or not.



```
But CC O O
its PRP$ O O
surprising JJ RH_NMOD O
progress NN O O
so RB O O
far RB LH_AMOD O
holds VBZ O ROOT
important JJ RH_NMOD O
lessons NNS O O
for IN LH_NMOD O
companies NNS LH_PMOD O
…………………
```

Figure 4: Sequential tagging model for neighbor parse

Similar to the neighbor parse, the root parsing task can also be treated as a sequential tagging problem. Figure 4 shows the basic concept of the root parser. The third column is mainly derived from the neighbor parser, while the fourth column represents whether the current word is a root with relation or not.

2.3 Parsing Algorithm

After adding the neighbor and root parser output as features, in the final stage, the modified Yamada's shift-reduce parsing algorithm (Yamada and Matsumoto, 2003) is then run. This method is deterministic and can deal with projective data only. There are three basic operation (action) types: Shift (S), Left (L), and Right (R). The operation is mainly determined via the classifier according to the selected features (see 2.4). Each time, the operation is applied to two unparsed words, namely, focus and next. If there exists an arc between the two words (either left or right), then the head of focus or next word is found; otherwise (i.e., shift), next two words are considered at next stage. This method could be economically performed via maintaining two pointers, focus, and next without an explicit stack. The parse operation is iteratively run until no more relation can be found in the sentence.

In 2006, Chang et al. (2006) further reported that the use of "step-back" in comparison to the original "stay". Furthermore, they also add the "wait-left" operations to prevent the "too early reduce" problems. In this way, the parse actions can be reduced to be bound in 3*n* where *n* is the number of words in a sentence.

Now we compare the adopted parsing algorithm in this year to the one we employed last year (Wu et al., 2006a). The common characteristics are:

1. the same number of parse operations (4)
2. shift-reduce
3. linearly scaled
4. deterministic and projective

On the contrary, their parse actions are quite different. Therefore these two methods have different run time. This gives the two methods rise to different iterative times. The main reason is that the step-back might trace back to previous words, which can be viewed as pop the top words on the stack back to the unparsed strings, while the Nivre's method does not trace-back any two words in the stack. In other words, if a word is pushed into the stack, it will no longer be compared with the other deeper words inside the stack. Hence some of the non-root words in the stack remain to be parsed. A simple solution is to adopt an exhaustive post-processing step for the unparsed words in the stack (details in (Wu et al., 2006a, 2006b)).

A good advantage of the step-back is that it can trace back to the unparsed words in the stack. But theoretically, the required parse actions still more than the Nivre's algorithm (2*n* vs. 3*n*).

By adopting the projectivized en/de-coding over the modified Yamada's algorithm, we can treat the words that do not have a parent as roots. Thus, for some languages (e.g. Czech and Arabic), the multiple root problem can be easily solved. In this year we separate the parse action and the relation label into two stages as opposed to having one pass last year. In this way, we can simply adopt a sequential tagger to auto-assign the relation labels after the whole sentence is parsed.

2.4 Features and Learners

Unlike last year, we did separate the action prediction and the label recognition into two stages where the one of the learners could provide more information to another. The used features of the two learners are quite similar and listed as follows:

Basic feature type (for previous 2 and next 3 words):
Word, POS (fine-grained), Lemma, FEAT, NParse, RParse

Enhanced feature type:
Bigram, BiPOS for focus and next words
previous two parse actions

For label recognition:
Label tag to its head, *label tags for previous two words*

In this paper, we replicate and modify the modified finite Newton support vector machines (MFN-SVM) (Keerthi and DeCoste, 2005) as the learner.

The MFN-SVM is a very efficient SVM optimization method which linearly scales with the number of training examples. Usually, the trained models from MFN-SVM are quite large that could not be processed in practice. We therefore defined the positive lower bound (10^{-10}) and the negative upper bound (-10^{-10}) to eliminate values that tend to be zero.

However, the SVM is a binary classifier which only recognizes true or false. For multiclass problem, we use the so-called one-versus-all (OVA) method with linear kernel to combine the results of each individual classifier. The final class in testing phase is mainly determined by selecting the maximum similarity.

For all languages, our parser uses the same settings and features. For all the languages (except for Basque and Turkish), we use backward parsing direction to keep the un-parsed token rate low.

3 Experimental Result

3.1 Dataset and Evaluation Metrics

The testing data is provided by the (Nivre et al., 2007) which consists of 10 language treebanks. More detailed descriptions of the dataset can be found at the web site[1]. The experimental results are mainly evaluated by the unlabeled and labeled attachment scores. CoNLL also provided a perl script to automatic compute these rates.

3.2 Results

Table 1 presents the overall parsing performance of the 10 languages. As shown in Table 1, we list two parsing results at column B and column C (new and old). It is worth to note that the result B is produced by training the neighbor parser with full labels instead of the three categories, left/right/out-of-neighbor. A is the official provided parse results. Some of the parsing results in A did not include the enhanced feature type and neighbor/root parses due to the time limitation. For the domain adaptation task, we directly use the trained English model to classify the PChemtb and CHILDES corpora without further adjustment.

In addition, we also apply the Maltparser 0.4, which is implemented with the Nivre's algorithm (Nivre et al., 2006) to be compared. The Maltpaser also includes the SVM and memory-based learner (MBL). Nevertheless, the training time complexity of the SVM in Maltparser is not linear time as MFN-SVM. Therefore we use the default MBL and feature model 3 (M3) in this experiment. To make a fair comparison, the input training data was also projectivized through the same pseudo-projective encoding/decoding methods.

[1] http://nextens.uvt.nl/depparse-wiki/SharedTaskWebsite

Table 1: A general statistical table of labeled attachment score, test and un-parsed rate (percentage)

Language	A	B	C	Statistic test			Un-Parsed Rate	
	(Official)	(Corrected)	(Malt-Parser 0.4)	A vs B	A vs C	B vs C	Old	New
Arabic	66.16	70.71	56.67	Yes	No	Yes	1.08%	0.69%
Basque	70.71	72.26	57.79	Yes	Yes	Yes	3.04%	3.72%
Catalan	81.44	81.44	76.36	Yes	No	No	0.45%	0.27%
Chinese	74.69	79.29	68.15	Yes	Yes	Yes	0.00%	0.00%
Czech	66.72	70.24	56.96	Yes	No	Yes	4.17%	3.87%
English	79.49	84.27	75.53	Yes	Yes	Yes	1.66%	0.84%
Greek	70.63	77.64	58.81	No	Yes	Yes	2.26%	2.12%
Hungarian	69.08	71.98	59.41	Yes	Yes	Yes	3.88%	5.38%
Italian	78.79	78.38	74.08	Yes	No	Yes	0.63%	0.63%
Turkish	72.52	75.65	64.41	Yes	Yes	Yes	4.93%	5.54%
pchemtb_closed	55.31**	73.35	-	-	-	-	-	-
*CHILDES_closed	52.89	58.29	-	-	-	-	-	-

* The CHILDES data does not contain the relation tag, instead, the unlabeled attachment score is listed

** The original submission of the pchemtb_closed task can not pass through the evaluator and hence is not the official score. After correcting the format problems, the actual LAS score should be 55.31.

To perform the significant test, we evaluate the statistical difference among the three results. If the answer is "Yes", it means the two systems are significant difference under at least 95% confidence score ($p < 0.05$).

The final column of the Table 1 lists the non-root words unparsed rate of the modified Yamada's method and the Nivre's parsing model which we employed last year. Among 10 languages, we can find that the modified Yamada's method outperform our old method in five languages, while fail to win in three languages. We did not report the comparative study between the forward parsing and backward parsing directions here since only the two languages (Basque and Turkish) were better in performing forward direction.

4 Discussion

Now we turn to discuss the improvement of the use of the neighbor parse and root parse. All of the experiments were conducted by additional runs where we removed the neighbor and root parse outputs from the feature set. In this experiment, we report four representative languages that tend to achieve the best and worst improvements. Table 2 lists the comparative study of the four languages.

As listed in Table 2, both English and Chinese got substantial benefit from the use of the two parsers. As observed by (Isozaki et al., 2004), incorporating both top-down (root find) and bottom-up (base-NP) can yield better improvement over the Yamada's parsing algorithm. Thus, instead of pre-determining the root and base-phrase structures, the tagging results of the neighbor and root parsers were included as new features to add wider information for the shift-reduce parser. It is also interesting to link neighbors and determine the root before parsing. We plan to compare it with out method in the future.

Table 2: The effective of the used Neighbor/Root Parser in the selected four languages

	With N/R Parser	Without
Chinese	79.29	75.51
English	84.27	79.49
Basque	72.26	72.32
Turkish	75.65	76.60

On the other hand, we also found that 2 out of the 10 languages had been negatively affected by the neighbor and root parsers. In Basque they made a marginally negative improvement, and in the Turkish the two parsers did decrease the original parsing models. We further observed that the main cause is that the weak performance of the neighbor parser. In Turkish, the recall/precision rates of the neighbor dependence are 92.61/93.12 with include neighbor parse outputs, while it achieved 93.71/93.51 with purely run the modified Yamada's method. We can expect that the result could achieve higher LAS score when the neighbor parser is improved. As mentioned in section 2.1, 2.2, the selected features for the two parsers are unified for the 10 languages. It is not surprising that for certain data the fixed feature set might perform even worse than the original shift-reduce parser. A better way is to validate the features with variant settings for different languages. We left the feature engine task as future work.

5 Conclusion and Future Remarks

Multilingual dependency parsing investigates on proposing a general framework of dependence parsing algorithms. This paper presents and analyzes the impact of two preprocessing components, namely, neighbor parsing and root-parsing. Those two parsers provide very useful additional features for downstream shift-reduce parser. The experimental results also demonstrated that the use of the two components did improve results for the selected languages. In the error-analysis, we also observed that for some languages, parameter tuning and feature selection is very important for system performance.

In the future, we plan to report the actual performance with replacing the MFN-SVM by the polynomial kernel SVM. In our pilot study, the use of approximate-polynomial kernel (Wu et al., 2007) outperforms the linear kernel SVM in Chinese and Arabic. Also, we are investigating how to convert the shift-reduce parser into approximate *N*-best parser efficiently. In this way, the parse reranking algorithm can be adopted to further improve the performance.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia and M. Oronoz. 2003. Construction of a Basque Dependency Treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajic, E. Hajicová and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, 103–127.

R. Brown. 1973. *A First Language: The Early Stages*. Harvard University Press.

M. W. Chang, Q. Do, and D. Roth. 2006. Multilingual Dependency Parsing: A Pipeline Approach. In *Recent Advances in Natural Language Processing*, pages 195-204.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang and Z. Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé (2003), chapter 13, pages 231–248.

Y. Cheng, M. Asahara and Y. Matsumoto. 2006. Multilingual Dependency Parsing at NAIST. *In Proc. of the 10th Conference on Natural Language Learning*, pages 191-195.

D. Czendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajic, O. Smrz, P. Zemánek, J. Snaidauf and E. Beska. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

H. Isozaki; H. Kazawa; T. Hirao. 2004. A Deterministic Word Dependency Analyzer Enhanced With Preference Learning. *In Proc. of the 20th International Conference on Computational Linguistics*, pages 275-281.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

S. Keerthi and D. DeCoste. 2005. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*. 6: 341-361.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

B. MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Lawrence Erlbaum.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez and M. Bertran. 2007. CESS-ECE: A Multilingual and Multilevel Annotated Corpus. Available for download from: http://www.lsi.upc.edu/~mbertran/cess-ece/.

R. McDonald, K. Lerman and F. Pereira. 2006. Multilingual Dependency Analysis with a Two-Stage Discriminative. *In Proc. of the 10th Conference on Natural Language Learning*, pages 216-220.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. *In Proc. of the International Workshop on Parsing Technology*, pages 149-160.

J. Nivre, and J. Nilsson. 2005. Pseudo-projective dependency Parsing. *In Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99-106.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. *In Proc. of the 10th Conference on Natural Language Learning*, pages 221-225.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek depen- dency treebank. *In Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

T. Kudo, K, Yamamoto, and Y. Matsumoto. 2004. Appliying conditional random fields to Japanese morphological analysis, *In Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 230-237.

Y. C. Wu, Y. S. Lee, and J. C. Yang. 2006a. The exploration of deterministic and efficient dependency parsing. *In Proc. of the 10th Conference on Computational Natural Language Learning*, pages 241-245.

Y. C. Wu, J. C. Yang, and Q. X. Lin. 2006b. Description of the NCU Chinese word segmentation and named entity recognition system for SIGHAN bakeoff 2006. *In Proc. of the 5th SIGHAN Workshop on Chinese Language Processing*, pages 209-212.

Y. C. Wu, J. C. Yang, and Y. S. Lee. 2007. An Approximate Approach for Training Polynomial Kernel SVMs in Linear Time. *In Proc. of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, in press.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. *In Proc. of the 8th International Workshop on Parsing Technologies*, pages 195–206.

Author Index